SYSTEMS AND METHODS FOR CREATING, MODIFYING, INTERACTING WITH AND PLAYING MUSICAL COMPOSITIONS

Field of the Invention

The present invention relates to systems and methods for creating, modifying, interacting with and playing music, and more particularly to systems and methods employing a top-down and interactive auto-composition process, where the systems/methods provide the user with a musical composition that may be modified and interacted with and played and/or stored (for later play) in order to create music that is desired by the particular user.

Background of the Invention

A large number of distinct musical styles have emerged over the years, as have systems and technologies for creating, storing, and playing back music in accordance with such styles. Music creation, particularly of any quality, typically has been limited to persons who have musical training or who have expended the time and energy required to learn and play one or more instruments. Systems for creating and storing quality musical compositions have tended towards technologies that utilize significant computer processing and/or data storage. More recent examples of such technologies include compact disc (CD) audio players and players of compressed files (for instance as per the MPEG-level 3 standard), etc. Finally, there exist devices incorporating a tuner, which permit reception of radio broadcasts via electromagnetic waves, such as FM or AM radio receivers.

Electronics and computer-related technologies have been increasingly applied to musical instruments over the years. Musical synthesizers and other instruments of increasing complexity and musical sophistication and quality have been developed, a "language" for conversation between such instruments has been created, which is known as the MIDI (Musical Instrument Digital Interface) standard. While MIDI-compatible instruments and computer technologies have

had a great impact on the ability to create and playback or store music, such systems still tend to require substantial musical training or experience, and tend to be complex and expensive.

Accordingly, it is an object of the present invention to provide systems and methods for creating, modifying, interacting with and/or playing music employing a top-down process, where the systems/methods provide the user with a musical composition that may be modified and interacted with and played and/or stored (for later play) in order to create music that is desired by the particular user.

It is another object of the present invention to provide systems and methods for creating, modifying, interacting with and/or playing music that enables a user to quickly begin creating desirable music in accordance with one or a variety of musical styles, with the user modifying an auto-composed or previously created musical composition, either for a real time performance and/or for storing and subsequent playback.

It is another object of the present invention to provide systems and methods for creating, modifying, interacting with and/or playing music in which a graphical interface is provided to facilitate use of the system and increase user enjoyment of the system by having graphic information presented in a manner that corresponds with the music being heard or aspects of the music that are being modified or the like; it also is an object of the present invention to make such graphic information customizable by a user.

It is another object of the present invention to provide systems and methods for creating, modifying, interacting with and/or playing music in which a graphical interface is provided that presents a representation of a plurality of musical lanes, below each of which is represented a tunnel, in which a user may modify musical parameters, samples or other attributes of the musical composition, with such modifications preferably being accompanied by a change in a visual effect.

It is another object of the present invention to provide systems and methods for creating, modifying, interacting with and/or playing music in which music may be represented in a form to be readily modified or used in an auto-composition algorithm or the like, and which presents reduced processing and/or storage requirements as compared to certain conventional audio storage techniques.

It is another object of the present invention to provide systems and methods for creating, modifying, interacting with and/or playing music in which music may be automatically composed in a variety of distinct musical styles, where a user may interact with auto-composed music to create new music of the particular musical style, where the system controls which parameters may be modified by the user, and the range in which such parameters may be changed by the user, consistent with the particular musical style.

It is another object of the present invention to provide systems and methods for creating, modifying, interacting with and/or playing music based on efficient song structures and ways to represent songs, which may incorporate or utilize pseudo-random/random events in the creation of musical compositions based on such song structures and ways to represent songs.

It is another object of the present invention to provide systems and methods for creating, modifying, interacting with and/or playing music in which songs may be efficiently created, stored and/processed; preferably songs are represented in a form such that a relatively small amount of data storage is required to store the song, and thus songs may be stored using relatively little data storage capacity or a large number of songs may be stored in a given data storage capacity, and songs may be transmitted such as via the Internet using relatively little data transmission bandwidth.

It is another object of the present invention to provide systems and methods for creating, modifying, interacting with and/or playing music in which a modified MIDI representation of music is employed, preferably, for example, in which musical rule information is embedded in MIDI pitch data, musical rules are applied in a manner that utilize relative rhythmic density and relative mobility of note pitch, and in which sound samples may be synchronized with MIDI events in a desirable and more optimum manner.

It is another object of the present invention to provide systems and methods for creating, modifying, interacting with and/or playing music in which a hardware/software system preferably includes a radio tuner so that output from the radio tuner may be mixed, for example, with auto-composed songs created by the system, which preferably includes a virtual radio mode of operation; it also is an object of the present invention to provide hardware that utilizes non-volatile storage media to store songs, song lists and configuration information, and hardware that

facilitates the storing and sharing of songs and song lists and the updating of sound banks and the like that are used to create musical compositions.

It is another object of the present invention to provide systems and methods for creating, modifying, interacting with and/or playing music that works in conjunction with a companion PC software program that enables users to utilize the resources of a companion PC and/or to easily update and/or share Play lists, components of songs, songs, samples, etc.

It is another object of the present invention to provide systems and methods for creating, modifying, interacting with and/or playing music in which songs may be generated, exchanged and disseminated, preferably or potentially on a royalty free basis.

Finally, it is an object of the present invention to provide systems and methods for creating, modifying, interacting with and/or playing music that may be adapted to a variety of applications, systems and processes in which such music creation may be utilized.

Summary of the Invention

The present invention addresses such problems and limitations and provides systems and methods that may achieve such objects by providing hardware, software, musical composition algorithms and a user interface and the like (as hereinafter described in detail) in which users may readily create, modify, interact with and play music. In a preferred embodiment, the system is provided in a handheld form factor, much like a video or electronic game. A graphical display is provided to display status information, graphical representations of musical lanes or components, which preferably vary in shape, color or other visual attribute as musical parameters and the like are changed for particular instruments or musical components such as a microphone input, samples, etc. The system preferably operates in a variety of modes such that users may create, modify, interact with and play music of a desired style, including an electronic DJ ("e-DJ") mode, a virtual radio mode, a song/song list playback mode, sample create/playback mode and a system mode, all of which will be described in greater detail hereinafter.

Preferred embodiments employ a top-down process, where the system provides the user with in effect a complete musical composition, basically a song, that may be modified and interacted with and played and/or stored (for later play) in order to create music that is desired by the particular user. Utilizing an auto-composition process employing musical rules and preferably a pseudo random number generator, which may also incorporate randomness

introduced by timing of user input or the like, the user may then quickly begin creating desirable music in accordance with one or a variety of musical styles, with the user modifying the auto-composed (or previously created) musical composition, either for a real time performance and/or for storing and subsequent playback.

A graphical interface preferably is provided to facilitate use of the system and increase user enjoyment of the system by having graphic information presented in a manner that corresponds with the music being heard or aspects of the music that are being modified or the like. An LCD display preferably is used to provide the graphical user interface, although an external video monitor or other display may be used as an addition or an alternative. In preferred embodiments, such graphic information is customizable by a user, such as by way of a companion software program, which preferably runs on a PC and is coupled to the system via an interface such as a USB port. For example, the companion software program may provide templates or sample graphics that the user may select and/or modify to customize the graphics displayed on the display, which may be selected and/or modified to suit the particular user's preferences or may be selected to correspond in some manner to the style of music being played. In one embodiment, the companion software program provides one or more templates or sample graphics sets, wherein the particular template(s) or sample graphic set(s) correspond to a particular style of music. With such embodiments, the graphics may be customized to more closely correspond to the particular style of music being created or played and/or to the personal preferences of the user.

The graphical interface preferably presents, in at least one mode of operation, a visual representation of a plurality of musical lanes or paths corresponding to components (such as particular instruments, samples or microphone input, etc.). In addition to allowing the user to visualize the various components of the musical composition, through user input (such as through a joystick movement) the user may go into a particular lane, which preferably is represented visually by a representation of a tunnel. When inside of a particular tunnel, a user may modify musical parameters, samples or other attributes of the musical composition, with such modifications preferably being accompanied by a change in a visual effect that accompany the tunnel.

In accordance with preferred embodiments, music may be automatically composed in a variety of distinct musical styles. The user preferably is presented with a variety of pre-set musical styles, which the user may select. As a particular example, in e-DJ mode, the user may select a particular style from a collection of styles (as will be explained hereinafter, styles may be arranged as "style mixes" and within a particular style mix one or more particular styles, and optionally substyles or "microstyles." After selection of a particular style or substyle, with a preferably single button push (e.g., play) the system begins automatically composing music in accordance with the particular selected style or substyle. Thereafter, the user may interact with the auto-composed music of the selected style/substyle to modify parameters of the particular music (such as via entering a tunnel for a particular component of the music), and via such modifications create new music of the particular musical style/substyle. In order to facilitate the creation of music of a desirable quality consistent with the selected style/substyle, the system preferably controls which parameters may be modified by the user, and the range over which such parameters may be changed by the user, consistent with the particular musical style/substyle. The system preferably accomplishes this via music that may be represented in a form to be readily modified or used in an auto-composition algorithm or the like. The musical data representation, and accompanying rules for processing the musical data, enable music to be auto-composed and interacted with in a manner that presents reduced processing and/or storage requirements as compared to certain conventional audio storage techniques (such as CD audio, MP3 files, WAV files, etc.).

In accordance with certain embodiments, the system operates based on efficient song structures and ways to represent songs, which may incorporate or utilize pseudo-random/random events in the creation of musical compositions based on such song structures and ways to represent songs. Songs may be efficiently created, stored and/processed, and preferably songs are represented in a form such that a relatively small amount of data storage is required to store the song. Songs may be stored using relatively little data storage capacity or a large number of songs may be stored in a given data storage capacity, and songs may be transmitted such as via the Internet using relatively little data transmission bandwidth. In preferred embodiments, a modified MIDI representation of music is employed, preferably, for example, in which musical

rule information is embedded in MIDI pitch data, and in which sound samples may be synchronized with MIDI events in a desirable and more optimum manner.

The system architecture of preferred embodiments includes a microprocessor or microcontroller for controlling the overall system operation. A synthesizer/DSP is provided in certain embodiments in order to generate audio streams (music and audio samples, etc.). Non-volatile memory preferably is provided for storing sound banks. Preferably removable non-volatile storage/memory preferably is provided to store configuration files, song lists and samples, and in certain embodiments sound bank optimization or sound bank data. A codec preferably is provided for receiving microphone input and for providing audio output. A radio tuner preferably is provided so that output from the radio tuner may be mixed, for example, with auto-composed songs created by the system, which preferably includes a virtual radio mode of operation. The system also preferably includes hardware and associated software that facilitates the storing and sharing of songs and song lists and the updating of sound banks and the like that are used to create musical compositions.

In alternative embodiments, the hardware, software, musical data structures and/or user interface attributes are adapted to, and employed in, a variety of applications, systems and processes in which such music creation may be utilized.

Such aspects of the present invention will be understood based on the detailed description to follow hereinafter.

Brief Description of the Drawings

The above objects and other advantages of the present invention will become more apparent by describing in detail the preferred embodiments of the present invention with reference to the attached drawings in which:

Fig. 1 illustrates an exemplary preferred embodiment of a "Player" in accordance with the present invention;

Figs. 2-3 illustrate exemplary preferred function and mode keys in accordance with the present invention;

Figs. 4-13 illustrate exemplary preferred screens of the graphical user interface in accordance with the present invention;

Fig. 14 is a table illustrating exemplary configuration parameters used in accordance with certain preferred embodiments of the present invention;

Fig. 15 illustrates the song structure used in certain preferred embodiments of the present invention;

Fig. 16 illustrates an exemplary preferred musical generation flow utilized in certain preferred embodiments of the present invention;

Fig. 17 is a table illustrating exemplary virtual notes/controllers utilized in certain preferred embodiments of the present invention;

Fig. 18 is a diagram illustrating Tessitura principles utilized in accordance with certain embodiments of the present invention;

Fig. 19 illustrates principles of encoding musical key changes preferably as offsets, which is utilized in accordance with preferred embodiments of the present invention;

Fig. 20 illustrates a mode application musical rule that preferably is part of the overall process in accordance with preferred embodiments of the present invention;

Fig. 21 illustrates an exemplary preferred virtual pattern to real pattern flow utilized in preferred embodiments of the present invention;

Fig. 22 illustrates principles of relative rhythmic density utilized in accordance with certain embodiments of the present invention;

Fig. 23 illustrates principles of the relative mobility of note pitch utilized in accordance with certain embodiments of the present invention;

Fig. 24 illustrates a pattern structure creation example in accordance with certain embodiments of the present invention;

Fig. 25 illustrates a block structure creation example in accordance with certain embodiments of the present invention;

Figs. 26-27 illustrate Pseudo-Random Number generation examples utilized in certain preferred embodiments of the present invention;

Fig. 28 illustrates attributes of simple data structures utilized in accordance with certain preferred embodiments of the present invention;

Fig. 29 illustrates an exemplary simple data structure flow in accordance with certain preferred embodiments of the present invention;

Fig. 30 illustrates attributes of complex data structures utilized in accordance with certain preferred embodiments of the present invention;

Fig. 31 illustrates an exemplary complex data structure flow in accordance with certain preferred embodiments of the present invention;

Figs. 32-34 illustrate exemplary hardware configurations of certain preferred embodiments of the player and a docking station in accordance with the present invention;

Fig. 35 illustrates an exemplary address map for the microprocessor utilized in accordance with certain preferred embodiments of the present invention;

Fig. 36 illustrates an exemplary address map for the synthesizer/DSP utilized in accordance with certain preferred embodiments of the present invention;

Figs. 37-38 illustrate the use of a DSP bootstrap/addressing technique utilized in accordance with certain preferred embodiments of the present invention;

Fig. 39 illustrates a simplified logical arrangement of MIDI and audio streams in the music generation process for purposes of understanding preferred embodiments of the present invention;

Fig. 40 illustrates a simplified MIDI and audio stream timeline for purposes of understanding preferred embodiments of the present invention; and

Figs. 41-42 illustrate the use of Non-Registered Parameter Number for purposes of synchronizing MIDA events and audio samples in accordance with certain preferred embodiments of the present invention.

Detailed Description of Exemplary Preferred Embodiments

The present invention will be described in greater detail with reference to certain preferred and certain other embodiments, which may serve to further the understanding of preferred embodiments of the present invention. As described elsewhere herein, various refinements and substitutions of the various elements of the various embodiments are possible based on the principles and teachings herein.

In accordance with the present invention, music may be created (including by auto-composition), interacted with, played and implemented in a variety of novel ways as will be hereinafter described via numerous exemplary preferred and alternative embodiments. Included in such embodiments are what may be considered as top-down approaches to musical creation.

Top-down as used herein generally means that a complete song structure for quality music is created for the end user as a starting point. This enables the user to immediately be in position to create quality music, with the user then having the ability to alter, and thereby create new music, based on the starting point provided by the system. Where a particular user takes the music creation process is up to them. More conventional musical creation processes involve a bottom-up approach, wherein the rudiments of each instrument and musical Style are learned, and then individual notes are put together, etc. This conventional approach generally has the side-effect of limiting the musical creation to a small group of trained people, and has, in effect, barred the wider population from experiencing the creative process with music.

A useful analogy for purposes of understanding embodiments of the present invention is that of building a house. In the conventional means of house-building, the user is given a bunch of bricks, nails, wood, and paint. If you want a house, you need to either learn all the intricacies of how to work with each of these materials, as well as electrical wiring, plumbing, engineering, etc., or you need to find people who are trained in these areas. Similarly, in musical creation, if you want a song (that is pleasing), you need to learn all about various types of musical instruments (and each of their unique specialties or constraints), as well as a decent amount of music theory, and acquire a familiarity with specific techniques and characteristics in a given Style of music (such as techno, jazz, hip-hop, etc.).

It would, of course, be far more convenient if, when someone wanted a house, they were given a complete house that they could then easily modify (with the press of a button). For example, they could walk into the kitchen and instantly change it to be larger, or a different color, or with additional windows. And they could walk into the bathroom and raise the ceiling, put in a hot tub, etc. They could walk into the living room and try different paint schemes, or different furniture Styles, etc. Similarly, in accordance with embodiments of the present invention, the user desirably is provided with a complete song to begin with, they can then easily modify, at various levels from general to specific, to create a song that is unique and in accordance with the user's desires, tastes and preferences.

In accordance with the present invention, the general population of people readily may be provided with an easy approach to musical creation. It allows them the immediate gratification of a complete song, while still allowing them to compose original music. This top down

approach to musical creation opens the world of musical creativity to a larger group of people by reducing the barriers to creating pleasurable music.

In accordance with the present invention, various systems and methods are provided that enable users to create music. Such systems and methods desirably utilize intuitive and easy to learn and use user interfaces that facilitate the creation of, and interaction with, music that is being created, or was created previously. Various aspects of one example of a preferred embodiment for a user interface in accordance with certain preferred embodiments of the present invention will now be described.

In accordance with such preferred embodiments of the present invention, user interface features are provided that desirably facilitate the interactive generation of music. The discussion of such preferred embodiments to be herein after provided are primarily focused on one example of a handheld, entry-level type of device, herein called 'Player'. However, many of the novel and inventive features discussed in connection with such a Player relate to the visual enhancement of the control and architecture of the music generation process; accordingly they can apply to other types of devices, such as computing devices, web server/websites, kiosks, video, or other electronic games and other entertainment devices that allow music creation and interaction, and thus also may benefit from such aspects of the present invention. A discussion of certain of the other types of devices is provided hereinafter. As will be appreciated by one of ordinary skill in the art, various features of the user interface of the Player can be understood to apply to such a broader range of devices.

Generally, the goal of the user interface is to allow intuitive, simple operation of the system and interaction with various parameters with a minimum number of buttons, while at the same time preserving the power of the system. Fig. 1 illustrates an exemplary system configuration for Player 10. Display 20 provides visual information to the user, as will hereinafter be described. Various mode keys 16 provide buttons that enable a user to directly access, or initiation, modes of operation of the system as will be hereinafter described. Joystick 15 is provided to enable the user to select or interact with various musical or system parameters or the like, as will be hereinafter described. Save/edit key 17 preferably is provided to save songs or parameter changes, etc., that a user may have created or made using the system, and also to initiate editing of parameters, Play lists, samples, etc., such as will be described hereinafter.

Volume key(s) 14 is/are provided, either in dual button up/down form or a single knob or dial to enable the output volume level to be adjusted. Function keys 11 preferably are provided to enable player functions such as play (ok), stop (cancel), forward (insert/create), reverse (delete) and record, exemplary uses of which will be described in greater detail hereinafter. FX key 12 preferably is provided to enable a user to easily and intuitively adjust one or more audio effects (e.g., doppler, reverb, wobbler, custom, etc.) of a part of the music (e.g., a particular sample sound); one preferred way to enable an intuitive sound effect selection by the user is to enable to FX key 12 to be used in combination with the Joystick 15 left and right controls, a corresponding preferred way to enable intuitive sound effect adjustment (e.g., increase or decrease the effect of the selected sound effect) is to enable to the FX Key 12 to be used in combination with the Joystick 15 up and down controls. Pitch/tempo key 13 preferably is provided to enable single button activation for pitch/tempo changes (preferably along with joystick movements), as will be hereinafter described in greater detail. On/off button 18 preferably is provided to turn on or off the player, and preferably a brief depression/toggle can be used to turn on/off an LCD backlight, although, for example, other turn off modes may be used as well (such as a time out turn off, when the player is not playing and there has been no activity detected for a predetermined time out period, etc. Exemplary desirable uses of such buttons and keys provided in the illustrative Player 10 embodiment will become more apparent based on the discussion to follow.

In accordance with preferred embodiments, a Home mode is provided. Home mode is a default mode that can be automatically entered when Player 10 is turned on. As the example of Fig. 4 shows, Home mode preferably displays an animated screen prompting the user to select a mode by pressing a direct access mode key 16 or entering help mode by pressing the joystick (Fig. 4 depicts the moment of the animation that prompts for the Radio direct access key). In preferred embodiments, a user can define the graphics displayed on the display 20 using, for example, a companion PC software program (discussed in greater detail below) to select graphics (animated or otherwise) to be automatically substituted (if available) for the default graphics during the different modes of operation. In this example of custom screens, data files corresponding to the customized screen graphics for each section of a song, and/or each mode of operation, preferably can be stored as part of the song data structure (discussed below) in a storage location of a removable memory means such as the Flash memory in a Smart Media Card

(SMC). In preferred embodiments, in Home mode the screen scrolls through various modes that are available in the system, such as modes associated with mode/direct access keys 16 (see, again, Fig. 1). Additionally, Player 10 preferably is configured to return to Home mode from the main menu of any other mode (i.e., from the user pressing the Stop key). When the joystick is pressed in Home mode, preferably a help screen is displayed prompting the user to press any key for help. An example help screen is shown in Fig. 5. In accordance with this example, when a key is pressed while Player 10 is displaying this screen, helpful text relating to that key is displayed.

Play can be used when in Home mode to enter a particularly important visual interface mode referred to herein as the I-Way mode (discussed in greater detail below). As shown in the example of Fig. 6, the preferably LCD screen can display a message regarding other possible modes, such as "e.DJ Style", in the status line and propose a selection of music Styles/SubStyles (e.g.; Techno Mix, House, Garage, etc.). At this type of screen, to select a desired Style, a user can press Up or Down. In this example, Styles in uppercase preferably denote a category of SubStyles that are randomly chosen for each song, and SubStyles preferably are indicated by lowercase Styles proceeding each uppercase Style. Once the user selects a Style, to enter I-Way mode with the selected Style, the user can press Play. Once the I-Way mode is entered, preferably Player 10 automatically creates, and starts playing, a song in the chosen Style. Exemplary Styles/SubStyles that preferably are provided in accordance with certain preferred embodiments include: Coolmix (SubStyles ballad, bossa, new age); Hip Hop Mix (SubStyles hip hop, rap, R&B, downbeat, ragga); Kitsch; Techno Mix (SubStyles house, garage, trance, jungle); etc. What is important to note is that, in accordance with preferred embodiments, distinct music Styles are determined, at least some of the musical Styles including distinct SubStyles, wherein characteristics of the particular Style and/or SubStyle result in different musical rules being applied to the automatic creation of music in accordance with the particular Style/SubStyle (the use of musical rules and other algorithmic and other details of the preferred music generation process is discussed in greater detail elsewhere herein), with an intuitive and easy to use interface provided to enable the ready creation and user modification of music in accordance with the particular Style/SubStyle, etc. In additional embodiments the use of an even finer gradation of musical aesthetic is available to the user in the form of a MicroStyle. For example, a plurality of

MicroStyles are provided that all generally conform to a particular SubStyle, while the SubStyle is accompanied by one or more other SubStyles that together generally conform to a particular Style. This third tier of musical granularity preferably gives the discerning user even finer control over the musical output of the algorithmic music. Such MicroStyles preferably provide more consistent music, while perhaps losing some of the flexibility of Styles/SubStyles. What is important is that the user is provided with a plurality of levels of musical style categorizations, where basically at each descending level the range of musical parameters that may be varied by the user and/or the auto-composition algorithm and the like are progressively more constrained, consistent with the particular Style, SubStyle or MicroStyle that is selected, etc.

An important feature of Home mode is the ability to configure Player 10 to start playing music quickly and easily. This is because, although Player 10 is configured to be interactive, and many professional-grade features are available to adjust various aspects of the Style and sound, it is desirable to have a quick and easy way for users to use the Player in a 'press-it-and-forget-it' mode. Thus, with only very few button pushes, a user with little or no musical experience, or little or no experience with Player 10, may easily begin composing original music with Player 10 of a desired Style or SubStyle. An additional preferred way to provide an auto-play type of capability is to use a removable storage memory medium (e.g., Smart Media Card) to store a Play list, such as a file containing a list of song data structures that are present on the removable memory. Following this example, when the user inserts the removable memory, or when the system is powered on with a removable memory already inserted, preferably the system will scan the removable memory to look for such a file containing a Play list and begin to play the song data structures that are listed in the system file. Preferably, this arrangement can be configured such that the Auto-Play mode is selectable (such as via a configuration setting in the system file), and that the system will wait a short duration before beginning Auto-Play, to allow the user an opportunity to enter a different mode on the system if so desired.

As illustrated in Fig. 7, an exemplary, preferred screen for an I-Way mode depicts the front view of the user driving or moving down a visual representation of a highway or multi-lane road or path. Along the very top of the screen preferably is a status message that displays the current section or status of the ongoing eDJ session (for example: part 1, filtering drums, chorus, Part 2, <<sample name>>, etc.). Preferably, other ways of displaying messages to the user to

more prominently indicate a status message can be used; for example, the system can momentarily flash a large visual indicator that takes up almost the entire screen. Preferably, directly in front of the field of view is a visual representation of a speaker that preferably is pulsing in time with the music being played. Preferably, each lane of the I-Way represents various types of elements of a song; such as instrument lanes (drums, bass, riff, lead), one or more sample lanes (to interact with pre-stored samples of voices, sounds, etc), and one or more microphone lanes which manage the microphone input in real-time. Other categories for lanes can be envisioned that are within the spirit and scope of the present invention. What is important to this aspect of the present invention that the user be presented with a multi-lane visual representation that includes a plurality of lanes, each of which corresponds to a constituent component or effect, etc., of the music that is being composed or played. The user preferably uses joystick 15 (for example, a circular button that can depress in 4 areas: top, bottom, left and right, such as illustrated in Fig. 1) to move the center of view around. Generally, each directional depression of joystick 15 causes the center of view to shift in the corresponding direction. For example, when in the left lane and the right joystick button is pressed, the center of view moves over one lane to the right. In alternative embodiments, additional layers of interactivity can be presented with additional horizontal layers of the I-Way. For example, when at the lane of the I-Way for the drums (an instrument with distinct instrument components, such as snare, bass, floor tom, high hat, crash cymbal, ping-ride cymbal, roto-toms, etc.; orchestral percussion, such as tympani, gong, triangle, etc.), the user could press the down key to go down to another I-Way for the drums or other multiple component instrument, with a lane for each drum or component, and/or for different aspects of the drum or instrument sound. This concept of multiple I-Way interfaces can be selectively used for only the instruments that benefit from such an approach, such as the drums or other multiple component instrument (while other instruments maintain a single I-Way interface, etc.). The use of additional I-Way lanes is not necessary to enjoy all the benefits of the present invention, but is a desirable feature for certain uses of the invention, such as products geared for more professional uses, or for music Styles where additional user interface and instrument control complexity is desirable, such as classical music, or jazz.

While in I-Way mode, the screen preferably is animated with sound waves or pulses synchronized with music beats. In the example of Fig. 7, a visual representation of a round

speaker is graphically represented in the center to symbolize the relative volume of the current lane. This graphic item preferably is configured to disappear, or be otherwise altered, when the lane is muted. It also can be configured to become bigger and smaller as the relative volume of that particular lane/section is adjusted (for example, by using a function key in combination with the joystick up and down buttons). Other simple variations are within the scope of the present invention, such as volume indicators visible in each lane at the same time, mute indications for each lane visible at the same time, graphic items in each lane visually reminiscent of the instrument represented by that lane, etc.

In an auto composition mode such as the I-Way mode it is Player 10 itself preferably that decides about a song progression in that it can automatically add/remove instruments, do music breaks, drums progressions, chord progressions, filtering, modulation, play samples in sync with the music, select samples to play based on rules, etc., to end up sounding like in a real song on a CD or from the radio. After a few minutes, if nothing is done by the user, Player 10 preferably is configured to end the song, preferably with an automatic fade out of volume, and automatically compose and play a new song in the same Style, or alternatively a different Style. It also should be understood that I-Way mode also is applicable in preferred embodiments for music that is not auto-composed, such as a song that the user created/modified using Player 10 (which may have been created in part using auto-composition) and stored in Player 10 for subsequent playback, etc.

In certain embodiments, newly composed patterns are numbered from 1 to n. This number can be displayed in the status line to help the user remember a music pattern he/she likes and come back to it after having tried a few other ones. In certain embodiments, this number might only be valid inside a given song and for the current interactive session. In other words, for example, the Riff pattern number 5 for the current song being composed would not sound like the Riff pattern number 5 composed in another song. However, if this song is saved as a user song, although the Riff music will be the same when replayed later, the number associated to it could be different.

In one exemplary embodiment, Player 10 "remembers" up to 16 patterns previously composed during the current interactive session. This means, for example, that if the current pattern number displayed is 25, the user can listen to patterns from number 10 to 25 by browsing

forward through the previously composed patterns (patterns 1-9, in this embodiment, having been overwritten or otherwise discarded). If the User wants to skip a given composed pattern that is currently being played, he/she can, and the pattern number will not be incremented, meaning that currently played pattern will be lost. This feature can be used to store only specific patterns in the stack of previously played patterns, as desired by the user. What is important is that the user can create musical patterns, and selectively store (up to some predetermined number of musical patterns), with the stored patterns used to compose music that is determined by the user based on the user's particular tastes or desires, etc. The views presented by I-Way mode desirably facilitate this user creation and interaction with, and modification of, the music that is be created/played by Player 10.

In certain preferred embodiments, if desired by a user, additional music parameters of an instrument associated with a particular lane in the I-Way mode may be "viewed" and interacted with by the user. For example, if a Down is pressed (such as by way of joystick 15) while in I-Way mode, the center of view is taken "underground," to the "inside" of a particular lane. This transition to Underground mode preferably is made visually appealing by configuring a screen animation depicting the movement of the point of view down through the floor or bottom of the I-Way lane, into what appears to be a visual representation of a tunnel below a particular lane that corresponds to the musical component represented by that lane. When inside the tunnel beneath a particular lane, a pulse indication (similar to the speaker pulse) preferably occurs in time with the tempo of the I-Way session. Furthermore, the left and right walls of the tunnel can be used to indicate the wave shape of the left and right sound channel outputs.

The far end of the tunnel preferably is comprised of a shape (for example, a rectangle or other geometric) that can change in correlation to the value of one or more of the parameters affecting the sound of that particular lane. By way of example, in the case of drums, a filter parameter can be changed by depressing the function or Fx button (see, again Fig. 1), plus the joystick up or down button; at this time the shape comprising the end of the tunnel either changes shape or visually appears to get farther away or nearer. In another example, the pitch of a guitar can be adjusted by pressing the pitch key along with the left or right joystick button; at the same time, the shape can become more or less slanted as the pitch parameter is incremented or decremented in value, or alternatively a visual representation of the tunnel going up hill or down

hill can be provided to visually represent an increase or decrease in pitch. In other examples, to change a right/left or stereo balance type of effect, the function or Fx button could be depressed to put the system in a mode to change the parameter along with left/right or up/down joystick button; such inputs could, for example, result in the sound balance going more towards the right channel than the left channel (and be accompanied by a visual representation of the tunnel turning to the right, or vice versa for the balance shifting towards the left channel), or the tunnel opening becoming larger in width or smaller in width if a wider or narrower stereo effect is desired. These are but several examples of how the shape or other visual effect can be modulated in correlation to the user input to one or more parameters effecting the sound. What is important is that, when the user "tunnels" into a particular instrument lane, various parameters associated with the instrument are changeable by the user, with at least certain of the changes in parameter being accompanied by a change in the visual representations provided to the user, such as the shape, size, color (for color display embodiments) or motions of the displayed visual representations.

While in Underground mode, Player 10 preferably is configured to continue looping with the same musical sequence while the user is able to interact with and modify the specific element (e.g., the drums) using the joystick and other buttons of Player 10. Also, while down in a lane corresponding to a particular component, preferably the left and right buttons of the joystick can be used to move from one component parameter to another. Alternatively, side to side joystick movements, for example, may enable the user to step through a series of preset characteristics or parameters (i.e., with simple joystick type user input, the user may change various parameters of the particular component, hear the music effect(s) associated with such parameter changes, and determine desirable characteristics for the particular music desired by the user at the particular point in time, etc.). In yet another alternative, side to side joystick movements, for example, may cause the view to shift from one tunnel to an adjacent tunnel, etc. All such alternatives are within the scope of the present invention.

In addition to other similar variations, the user can mute a particular lane in the I-Way mode preferably by use of Stop key (shown in Fig. 2). In this example, while the lane is muted, "Muted" can be displayed in the status bar and the round speaker can disappear. Preferably in

accordance with such embodiments, the user can un-mute the instrument by again pressing the Stop key.

An additional desirable variation of the user interface preferably involves animating a change to the visual appearance, corresponding to a new song part. For example, if in the Underground mode shown in Fig. 8, or in the I-Way mode shown in Fig. 7, the movement to a chorus section is accompanied by a movement through an opening doorway. The graphic animation corresponding to a given section of the song (e.g., chorus, intro, bridge, ending, etc.) can be used each time that section is played during the song. Examples of transitions are: having the user go through a door from a tunnel with one set of visual characteristics, to a tunnel with a second set of visual characteristics. Another example is to have the user move through a transition doorway from a tunnel to a wider tunnel, or even an open area. The preferable feature of this aspect of the present invention is to provide an engaging experience for the user by coordinating an animation transition that is closely linked to a musical transition between song parts.

Alternatives to the I-Way and Underground concepts can also be advantageously used with the present invention. For example, a user interface that visually depicts the instruments that are in the current song, and allows the user to select one to go into a tunnel or level where parameters of the particular instrument may be adjusted. In this example, while the music is playing, the user interface provides visual representations of the instruments in the current song, with the active instruments preferably emitting a visual pulse in time with the music. Fig. 13 is an example of such a user interface. In accordance with such embodiments, the user can select a particular visual picture of an instrument (for example, such as with joystick 15 or function keys 11) and go into that instrument. For example, by selecting the vibrating drumset 25, the user can go into another level, such as corresponding to the Underground mode discussed above with reference to Fig. 12, that has each drum shown that is currently being played. Then, the user can select and change different aspects of the drums, as well as the sound effects, and drum tracks. If the user selected another instrument such as are shown in Fig. 13, they would access a screen that allows them to similarly alter the parameters of that particular instrument track. Accordingly, the use of alternative themes for the user interface can be advantageously employed with the present invention, especially a theme where the actual instruments are depicted, as if on a stage. In

certain embodiments, both or multiple types of user interfaces are provided, and the user may select an I-Way type of user interface, such as shown in Fig. 7, or instrument group or other type of interface. What is important is that the user interface in preferred embodiments preferably provide an intuitive and easy to use way for users, who may have little experience in creating music, to visually appreciate the instruments used to create the music, and then have a visual way to access a mode in which parameters and effects associated with particular instruments may be modified by the user, which is preferably accompanied by a visual change that corresponds to the modified parameters/effects, etc.

Additionally, in certain preferred embodiments, the use of an external video display device (e.g., computer monitor, television, video projector, etc.) is used to display a more elaborate visual accompaniment to the music being played. In such cases the I-Way graphical display preferably is a more detailed rendition of the I-Way shown in Fig. 7 (e.g., a higher resolution image in terms of color depth and/or dots per inch).

In certain preferred embodiments, pressing Play preferably causes the lane instrument to enter Forced mode. This can be implemented to force Player 10 to play this instrument pattern at all times until Forced mode is exited by pressing Play again when the lane of that instrument is active. In this case, if the instrument was not playing at the time Forced mode is selected, Player 10 can be configured to automatically compose the instrument pattern and play it starting at the end of the current sequence (e.g., 2 bars). In addition, pressing Play for a relatively long period (e.g., a second or more) can pause the music, at which time a "paused" message can flash in the status line.

In other preferred embodiments, where such a Forced mode may not be desired (e.g., for simplicity, and/or because it may not be needed for a particular type of music), pressing Play briefly preferably causes a Pause to occur. Such a pause preferably would have a 'Paused' message appear on the Display 20, and preferably can be rhythmically quantized such that it begins and ends in musical time with the song (e.g., rhythmically rounded up or down to the nearest quarter note).

Solos

In Solo mode, all other instruments are muted (except for those that may already be in Solo mode) and only this instrument is playing. Solo mode preferably is enabled by entering a

tunnel or other level for a particular instrument, and, if the instrument is already playing entering Solo mode upon pressing of Play (e.g., the instrument is in Forced play and subsequent pressing of Play in Underground mode initiates Solo mode for that instrument; the particular key entry into Solo mode being exemplary). An instrument preferably remains soloed when leaving the corresponding tunnel and going back to the music I-Way. The user also preferably must re-enter the corresponding tunnel to exit Solo mode. Also, in certain embodiments multiple levels of Solo mode are possible in that you can solo several tracks, one at a time or at the same time, by going into different tunnels and enabling Solo mode. In addition, in certain embodiments the user preferably can enable/disable Solo mode from the I-Way by, for example, pressing Play for a long time (e.g., 2 seconds) while in a lane. Following this example, upon disabling Solo mode, any lanes that had previously been manually muted (before Solo mode was invoked) preferably will remain muted.

Preferably, from a Sample menu different sample parameters can be edited. From the Samples menu, the user can record, play and change effects on voice, music or sound samples. This menu also preferably permits the creation and edition of sample lists. The LCD preferably displays "e.Samples" in the status line and a list of available samples or sample lists in the storage media (for example, the SmartMedia card, discussed in connection with Fig. 32) to choose from.

When playing back a sample, the LCD preferably displays the play sample screen. The name of the sample preferably scrolls in a banner in the center right part of the LCD while the audio output level is indicated by a sizable frame around the name. The status line preferably shows the current effect.

Sample sets or lists preferably are used by the e.DJ, for user songs, as well as MIDI files. In the case of MIDI files, preferably a companion PC software program (e.g., a standard MIDI editing software program such as Cakewalk) is used to enable the user to edit their own MDI files (if desired), and use MIDI non-registered parameter numbers (NRPNs are discussed below in more detail) to effectuate the playing of samples at a specific timing point. Following this example, the companion PC software program can be enabled to allow the user to insert samples into the MIDI data, using NRPNs. When a new e.DJ song is created, Player 10 preferably picks one of the existing sample lists (sample sets preferably being associated with the particular

Style/SubStyle of music) and then plays samples in this list at appropriate times (determined by an algorithm, preferably based on pseudo random number generation, as hereinafter described) in the song. When creating or editing a user song, the user preferably can associate a sample list to this user song. Then, samples in this list will be inserted automatically in the song at appropriate times. Each sample list can be associated with an e.DJ music Style/SubStyle. For instance, a list associated with the Techno Style can only be used by a Techno user song or by the e.DJ when playing Techno Style. In additional variations, the user preferably can specify specific timing for when a particular sample is played in a song, by way of NRPNs discussed below. This specification of the timing of a particular sample preferably can be indicated by the user through the use of a companion PC software program (e.g., a standard MIDI editing software program such as Cakewalk), and/or through a text interface menu on the Player 10 itself.

New Sample lists preferably are created with a default name (e.g., SampleList001). The list preferably can be renamed in the System-files menu. When the selected item is a sample, the current effect preferably is displayed in the status line. When the selected item is a sample list, "List" preferably is displayed in the status line.

Playback of preferably compressed audio, MIDI, Karaoke, and User songs (e.g., e.DJ songs that have been saved) preferably is accessible via the "Songs" mode. Songs can be grouped in so-called Play lists to play programs (series) of songs in sequence. The LCD will display "e.Songs" in the status line and a list of available songs or Play lists on the SmartMedia card to choose from.

Depending on the type of the song (for example, user song, MIDI or WMA), different parameters can be edited. The type of the current selection preferably is indicated in the status bar: WMA (for WMA compressed audio), MID (for MIDI songs), KAR (for MIDI karaoke songs), MAD x (for user songs {x=T for Techno Style, x=H for Hip-Hop, x=K for Cool, etc.}), and List (for Play lists).

The name of the song preferably scrolls in a banner in the center right part of the LCD while the audio output level is indicated by a sizable frame around the name. If the song is a karaoke song, the lyrics preferably are displayed on two (or other number) lines at the bottom of the LCD. The animated frame preferably is not displayed. If the song is a user song (i.e.,

composed by the e.DJ and saved using the Save/Edit button), the music I-Way mode is entered instead of the play song mode.

The edit screen preferably is then displayed, showing two columns; the left column lists the editable parameters or objects in the item, the right column shows the current values of these parameters. For example, a Play list edit screen preferably will display slot numbers on the left side and song names on the right side. The current object preferably is highlighted in reverse video.

Play lists are used to create song programs. New Play lists are preferably created with a default name (e.g., PlayList001), and preferably can be renamed by the user. When a list is selected and played in the song select screen, the first song on the list will begin playing. At the end of the song, the next song preferably will start and so on until the end of the list is reached. Then, if the terminating instruction in the list is End List, the program preferably stops and Player 10 returns to the song select screen. If the terminating instruction is Loop List, the first song preferably will start again and the program will loop until the user interrupts the song playing, such as by pressing the stop button.

In one embodiment of the present invention, the features of a conventional radio are effectively integrated into the user interface of the present invention (see, e.g., the FM receiver 50 of Fig. 32). For example, when playing a station in Radio mode, the LCD preferably will display a radio screen. The LCD preferably will display "Radio" in the status line as well as a list of available station presets to choose from. If no preset has been preset, only the currently tuned frequency might be displayed. The name of the radio station (or frequency if it is not a stored preset) can scroll in a banner in the center right part of the LCD. An animation representing radio waves can also be displayed. The status line preferably shows the tuned frequency. In such embodiments Player 10 is enabled to operate as a conventional radio device.

In preferred embodiments, radio-type functionality involves the use of the same type of Radio interface, with virtual stations of different Styles. Each virtual station preferably will generate continuous musical pieces of one or more of a particular Style or SubStyle. In this v.Radio mode, the user can "tune-in" to a station and hear continuous music, without the use of an actual radio. Such an arrangement can provide the experience of listening to a variety of music, without the burden of hearing advertising, etc., and allows the user to have more control

over the Style of music that is played. In such embodiments, a user will enter v.Radio mode and be presented with a list of v.Radio stations, each preferably playing a particular Style or SubStyle of music. The user then preferably "tunes" to a v.Radio channel by selecting a channel and pressing play, for example (see, e.g., Fig. 10), which causes Player 10 to begin auto-composing and playing songs in accordance with the particular v.Radio channel. In certain embodiments, the v.Radio may be controlled to play user songs of the particular Style or SubStyle associated with the particular v.Radio channel, which may be intermixed with auto-composed songs of the particular type of SubStyle. In yet other embodiments, one or more v.Radio channels may be provided that play songs of more than a single Style or SubStyle, which also may be intermixed with user songs of various Styles or SubStyles. With such embodiments, the user is provided options to select the particular type of v.Radio channel that Player 10 "tunes" in. Additionally, in certain embodiments the v.Radio mode preferably can be used to play a variety of different song formats (e.g., MP3, WAV, WMA, eDJ, etc.).

In accordance with certain embodiments, another variation of the Radio feature integrates some aspects of the v.Radio with other aspects of the Radio. As one example, a user could listen to a Radio station, and when a commercial break comes on, Player 10 switches to the v.Radio. Then, when the real music comes back on, the device can switch back to a Radio. Another integration is to have news information from the Radio come in between v.Radio music, according to selectable intervals. For example, most public radio stations in the USA have news, weather, and traffic information every ten minutes during mornings and afternoons. The v.Radio can be configured to operate as a virtual radio, and at the properly selected interval, switch to a public station to play the news. Then it can switch back to the v.Radio mode. These variations provide the capability for a new listening experience, in that the user can have more control over the radio, yet still be passively listening. It is considered that such an arrangement would have substantial use for commercial applications, as discussed elsewhere in this disclosure.

Special functions can preferably be accessed from the System menu. These functions preferably include: file management on the SmartMedia card (rename, delete, copy, list, change attributes) (the use of such SmartMedia or other Flash/memory/hard disk type of storage medium is discussed, for example, in connection with Fig. 32), Player configuration (auto-play, power off, delay, keypad auto-repeat, language, etc.), firmware upgrade, SmartMedia card formatting,

microphone settings, and equalizer user presets. The Player can preferably modify various attributes of a file stored on the SmartMedia card. As a precaution, by default, all system files preferably can be set as read only.

In certain embodiments a User Configuration interface preferably enables the user to enter a name to be stored with the song data on the removable memory storage (e.g., SMC), and/or to enable the user to define custom equalization settings, and/or sound effects. As an example of EQ settings, it is preferable to enable the user to select from a group of factory preset equalizer settings, such as flat (e.g., no EQ effect), standard (e.g., slight boost of lower and higher frequencies), woof (e.g., bass frequency boost), and hitech (e.g., high frequency boost). In addition to such preset EQ settings, it is preferable to enable the user to define their own desired settings for the EQ (as an example, a 4 band EQ with the ability to adjust each of the 4 bands by way of the joystick). Additionally, in certain embodiments it is preferable to enable the user to similarly customize sound effects to be used for particular samples. Following this example, in addition to a set of standard factory preset sound effects such as Lowvoice (e.g., plays the song with a slower speed and lower pitch to enable the user to sing along with a lower voice), reverb, Highvoice (e.g., plays the song with a faster speed and higher pitch), Doppler (e.g., varying the sound from Highvoice to Lowvoice), and Wobbler (e.g., simulating several voices with effects), it is preferable to make a customized effect capability available to the user that can incorporate various combinations of standard effects, and in varying levels and/or with varying parameter values.

When the user saves a song that is being played in e-DJ mode, the song is preferably created with a default name (e.g. TECHNO001). The song can preferably be renamed in the System-files menu. When entering the Files menu, files present on the SmartMedia card and the free memory size are preferably listed in an edit screen format. The status line preferably indicates the number of files and the amount of used memory. The file management menu preferably offers a choice of actions to perform on the selected file: delete, rename, copy, change attributes, etc. The name of the current file preferably is displayed in the status line. Additionally, in certain embodiments it is preferable to enable the use of System parameter files that contain, for example, settings for radio presets (e.g., radio station names and frequencies), settings for certain parameters (e.g., pitch, tempo, volume, reverb, etc.) associated with music

files such as WAV, WMA, MP3, MIDI, Karaoke, etc. In these embodiments it is preferable for the parameter setting to apply to the entire file.

When entering the Configuration menu, an edit screen preferably is displayed showing various configurable parameters. Fig. 14 describes some of the parameters that are preferably configurable by the Configuration menu, along with possible values. When modifying a selected character in a file name, Forward preferably can be used to insert a character after the highlighted one, and Backward preferably to delete the highlighted character. To save the edits and go back to file menu, Play preferably can be used.

When selecting copy, a screen proposing a name for the destination file in a large font preferably is displayed. This name preferably is proposed automatically based on the type of the source file. For instance if the source file is a Hiphop user song, the proposed name for the destination file could be HIPHOP001 (alternatively, the user preferably can use the rename procedure described above to enter the name of the destination file).

The Firmware Upgrade menu preferably permits the upgrade of the Player firmware (embedded software) from a file stored on the SmartMedia card. Preferably, it is not possible to enter the Upgrade firmware menu if no firmware file is available on the SmartMedia card. In this case a warning message is displayed and the Player preferably returns to Systems menu. In additional embodiments, the use of a bootstrap program preferably is enabled to allow the firmware to be updated from a removable memory location (e.g., SMC). Such a bootstrap program preferably can alternatively be used for upgrading the DSP 42 soundbank located in Flash 49.

The Player function keys, identified in Fig. 2, preferably are comprised of the standard buttons found on CD-players or VCRs, and are used to control the playback of songs (e.g.; Player-proprietary, MIDI, WMA, MP3, etc). The Record key controls recording (e.g.; samples). When used in editing or selection menus the player keys also have the following actions: Play preferably is used to select a sub menu or validate a change, Stop preferably is used to go back to previous menu, cancel an action or discard a change, Forward preferably is used to insert an item in a list, and REVERSE preferably is used to delete an item in a list. This is one example of how to use a minimum of keys in a device, while retaining a relatively large set of features, while also keeping the user interface relatively intuitive for a variety of users.

When a list is selected in the song select screen, pressing Play preferably will start playing the first song in the list. While the sample lane is selected, Play preferably can be configured to start playing the selected sample. While in an instrument lane, Play preferably can be configured to enter solo mode for the current instrument, or Forced mode.

To create a song/sample list, Forward preferably can be used while in the song or sample select screen.

To leave an edit screen, Stop preferably can be used to discard the edits and exit. For example, in the sample selection screen press Stop to go back to the Home screen. Additionally, for any given instrument during playback, Stop preferably can be used as a toggle to mute/unmute the instrument.

Record preferably can be pressed once to start recording a sample (recording samples preferably is possible in almost any operating mode of the Player). Record preferably can be pressed again to end the recording (recording preferably is stopped automatically if the size of the stored sample file exceeds a set size, such as 500Kbytes). The record source preferably is chosen automatically depending on the operating mode. If no music is playing, the record source preferably is the active microphone (local or docking station). If music is playing songs, e.DJ or radio, the record source preferably is a mix of the music and the microphone input if not muted. Further, it is possible to use different sample recording formats that together provide a range of size/performance options. For example, very high quality sample encoding format may take more space on the storage medium, while a relatively low quality encoding format may take less space. Also, different formats may be more suited for a particular musical Style, etc.

In v-Radio mode, to listen to the selected station, Play preferably can be used. Press Play to mute the radio. Press Stop to go back to station preset selection screen. To launch an automatic search of the next station up the band, press Forward until the search starts. To launch an automatic search of the next station down the band, press Backward until the search starts. Press Forward/Backward briefly to fine-tune up/down by 50kHz steps.

In eDJ Mode, while in Sample lane, Play preferably can be pressed to play a selected sample. If sample playback had previously been disabled, the first press on Play preferably will re-enable it. Subsequent presses preferably will play the selected sample. If a sample if playing, Stop preferably will stop it. If no sample is playing, pressing Stop preferably will mute the

samples (i.e. disable the automatic playback of samples by the e-DJ when returning to I-Way mode). When muted, "Muted" preferably is displayed in the status bar and the round speaker preferably disappears on the I-Way sample lane.

In Song mode, to start the playback of selected song or Play list, preferably press Play and the LCD will preferably display the play song screen. In Song mode, Stop preferably can be pressed to stop the music and preferably go back to song selection screen. Preferably press Forward briefly to go to next song (if playing a Play list, this preferably will go to the next song in the list; otherwise, this preferably will go to the next song on the SmartMedia). Preferably press Forward continuously to fast forward the song. Preferably press Backward briefly to go to the beginning of the song and a second press preferably takes you to the previous song (if playing a Play list, this preferably will go to the previous song in the list; otherwise, this preferably will go to the previous song on the SmartMedia). Preferably press Backward continuously to quickly go backward in the song.

Pressing Stop can be a way to toggle the muting of an instrument/lane. For example, when on a Drums lane, pressing Stop briefly preferably can mute the drums, and pressing it again briefly preferably can un-mute the drums. Additionally, pressing Stop for relatively long period (e.g., a second or so) preferably can be configured to stop the music and go back to Style selection screen.

Forward preferably can be configured to start a new song. Backward preferably can be used to restart the current song.

Forward or Backward preferably can be used to keep the same pattern but change the instrument playing (preferably only "compatible" instruments will be picked and played by the Player).

Preferably press Stop to mute microphone. Preferably press Play to un-mute the microphone.

To start the playback of the selected sample, preferably press Play. Preferably press Stop to stop the sample and go back to sample selection screen.

In Song mode, preferably press Play to pause the music. Preferably press Play again to resume playback. Pressing Forward key in the song select screen preferably will create a new Play list. In the song selection screen, preferably press Stop to go back to the Home screen.

In the Style selection screen preferably press Stop to go back to the Home screen.

To enter the file management menu for the highlighted file, preferably press Play.

While browsing the file management list, preferably press Forward to scroll down to next page. Press Backward preferably to scroll up to previous page.

In the file management menu, to start a selected action, preferably press Play.

When selecting Delete, preferably a confirmation screen is displayed.

When selecting Rename, preferably a screen showing the name of the file in big font is displayed and the first character is preferably selected and blinking.

When copying a file, preferably press Play to validate the copy. If a file of the same type as the source file exists with the same name, preferably a confirmation screen asks if the file should be overwritten. Select YES or No and preferably press Play to validate. Press Stop to abort the copy and preferably return to file menu. It is a preferable feature of this embodiment to allow files to be copied from one removable memory storage location (e.g., SMC) to another by use of MP 36 RAM. In this example, it is a desirable to enable the copying of individual song or system files from one SMC to another without using a companion PC software program, however, in the case where an entire removable memory storage volume (e.g., all the contents of a particular SMC) is to be copied, it is desirable to use a companion PC software program to allow larger groups of data to be temporarily buffered (using the PC resources) by way of the USB connection to the PC. Such a feature may not be possible in certain embodiments without the PC system (e.g., using the MP 36 internal RAM) because it likely would involve the user repeatedly swapping the SMC target and source volumes.

The e-DJ, v-Radio, Songs, Samples and System direct access keys detailed in Fig. 3 preferably permit the user to directly enter the desired mode from within any other mode. These keys preferably can also be used to stop any mode, including the current mode. This can be faster than the Stop key, because in some cases, such as while in eDJ Mode inside a lane, the Stop key preferably may be used to mute the lane, rather than stop the eDJ Mode.

The audio output control is identified in Fig. 1 as Vol. Up/Down. Audio output control keys preferably are also used to control the microphone input when used in combination with prefix keys.

The Up/Down/Left/Right keys preferably comprise a joystick that can be used for: menu navigation, song or music Style selection, and real time interaction with playing music. Additionally, Up/Down preferably can be used for moving between modes such as the Underground & I-Way modes in an intuitive manner.

When editing a list, objects preferably can be inserted or deleted by pressing Forward to insert an object after the highlighted one or pressing Backward to delete the highlighted object.

To browse the list or select parameters, preferably use Up/Down. To edit the highlighted object preferably press Right. Press Left preferably to go directly to first item in the list.

In instrument tunnels (i.e.; Drums, Bass, Riff and Lead), Right preferably can be configured to compose a new music pattern. Similarly, Left preferably can be used to return to previous patterns (see note below on music patterns). The new pattern preferably will be synchronized with the music and can start playing at the end of the current music sequence (e.g., 2 bars). In the mean time, preferably a "Composing..." message can be configured to appear on the status line. Additionally, Down preferably can be used to compose a new music pattern without incrementing the pattern number. This preferably has the same effect as Right (compose and play another pattern), except that the pattern number preferably won't be incremented.

One benefit of these composition features is that they enable the user to change between patterns during a live performance. As can be appreciated, another reason for implementing this feature is that the user preferably can assemble a series of patterns that can be easily alternated. After pressing Right only to find that the newly composed pattern is not as desirable as the others, the user preferably can subsequently select Down to discard that pattern and compose another. Upon discovering a pattern that is desirable, the user preferably can thereafter use Right and Left to go back and forth between the desirable patterns. Additionally, this feature preferably allows the system to make optimum use of available memory for saving patterns. By allowing the user to discard patterns that are less desirable, the available resources preferably can be used to store more desirable patterns.

In the file management menu, to select a desired action, preferably use Up/Down. When renaming files, the user preferably can use Left/Right to select the character to be modified, and Up/Down to modify the selected character. Pressing Right when the last character is selected

preferably will append a new character. The user preferably can also use the Forward/Backward player function keys at these times to insert/delete characters.

In the microphone tunnel, Left/Right preferably can be configured to change microphone input left/right balance. In the sample tunnel, Left/Right preferably can be used to select a sample. Pressing Forward in the sample select screen preferably will create a new sample list.

Down is an example of an intuitive way to enter the Underground mode for the current I-Way mode lane. In this mode, the user preferably can change the pattern played by the selected instrument (drums, bass, riff or lead) and preferably apply digital effects to it. Similarly, Up preferably can be configured to go back to music I-Way from the Underground mode.

In v-Radio mode, to select the desired station preset, preferably use Up/Down. Preferably use Up/Down to go to previous/next station in the preset list and preferably press Save/Edit while a station is playing to store it in the preset list.

The Save/Edit key preferably can be used to save the current song as a User song that can be played back later. Such a song preferably could be saved to a secondary memory location, such as the SmartMedia card. In the case of certain Player embodiments, this preferably can be done at any time while the e-DJ song is playing, as only the "seeds" that generated the song preferably are stored in order to be able to re-generate the same song when played back as a User song. In certain embodiments it is preferable to incorporate a save routine that automatically saves revised files as a new file (e.g., with the same name but a different suffix). Such a feature can be used to automatically keep earlier versions of a file.

While the use of seeds is discussed elsewhere in this disclosure, it may be helpful at this point to make an analogy on the use of the Save/Edit 17 key. This key is used to save the basic parameters of the song in a very compact manner, similar to the way a DNA sequence contains the parameters of a living organism. The seeds occupy very little space compared to the information in a completed song, but they are determinative of the final song. Given the same set of saved seeds, the Player algorithm of the present invention preferably can generate the exact same sequence of music. So, while the actual music preferably is not stored in this example (upon the use of the Save/Edit 17 key), the fundamental building blocks of the music is stored very efficiently. The desirability of such an approach can be appreciated in a system with relatively limited resources, such as a system with a relatively low-cost/low performance

processor and limited memory. The desirability of such a repeatable, yet extremely compact method of storing music can also be contemplated in certain alternative embodiments, such as those involving the communication with other systems over a relatively narrow band transmission medium, such as a 56kbps modem link to the internet, or an iRDA/bluetooth type of link to another device. Clearly this feature can be advantageously employed using other relatively low bandwidth connections between systems as well. Additionally, this feature allows the user to store many more data files (e.g., songs) in a given amount of storage, and among other advantages, this efficiency enhances other preferable features, such as the automatic saving of revised files as new files (as discussed above).

In certain embodiments, it is desirable to check the resources available to a removable memory interface (e.g., the SMC interface associated with SMC 40) to safeguard the user song in instances where a removable memory volume is not inserted, and/or there is not enough available storage on an inserted removable memory volume. In these cases, when the user saves a song (e.g., pushes the Save/Edit key 17 button) it is advantageous to prompt the user to insert an additional removable memory volume.

The name of the song preferably can be temporarily displayed in the status line, in order to be able to select this song (as a file) later on for playback. Of course the song file name preferably can be changed later on if the User wishes to do so. Once an item has been created, it preferably can be edited by selecting it in the song or sample selection screens and pressing Save/Edit. Pressing Save/Edit again will preferably save the edited item and exit. When the On/Off key is pressed for more than 2 seconds, the Player preferably can be configured to turn on or off, yet when this combination is pressed only briefly, the On/Off key can alternatively preferably be configured to turn the LCD backlight on or off.

When Pitch/Tempo is pressed simultaneously with Left or Right, it preferably can be used as a combination to control the tempo of the music. When Pitch/Tempo is pressed simultaneously with Up/Down, it preferably can control the pitch of the microphone input, the music, etc.

When Effects/Filters is pressed simultaneously with Left/Right or Up/Down, it preferably can control the effect (for example, cutoff frequency or resonance) and/or volume (perhaps including mute) applied on a given instrument, microphone input, or sample.

As will be appreciated by one of ordinary skill in the art, other related combinations can be employed along these lines to provide additional features without detracting from the usability of the device, and without departing from the spirit and scope of the present invention.

Various examples of preferred embodiments for the structuring of a song of the present invention will now be described. Preferably for a new song, the only user input needs to be an input Style. Preferably even this is not required when an auto-play feature is enabled that causes the Style itself to be pseudo-randomly selected. But assuming the user would like to select a particular Style, that is the only input preferably needed for the present embodiment to begin song generation.

Before moving into the actual generation process itself, it is important to note that preferably implicit in the user's Style selection can be a Style and a SubStyle. That is, in certain embodiments of the present invention, a Style is a category made up of similar SubStyles. In these cases, when the user selects a Style, the present embodiment will preferably pseudo-randomly select from an assortment of SubStyles. Additionally, it is preferably possible for the user to select the specific SubStyle instead, for greater control. In these particular embodiments, preferably whether the user selects a Style or a SubStyle, the result preferably is that both a Style and a SubStyle can be used as inputs to the song generation routines. When the user selects a SubStyle, the Style preferably is implicitly available. When the user selects a Style, the SubStyle preferably is pseudo-randomly selected. In these cases, both parameters are available to be used during the song generation process to allow additional variations in the final song.

As shown in Fig. 15, the Song is preferably comprised of a series of Parts. Each part preferably might be an intro, theme, chorus, bridge, ending, etc.; and different parts preferably can be repeated or returned to later in a song. For example, one series of parts might be: intro, theme, chorus, theme, chorus, theme, chorus, end. Certain Styles preferably may have special types of parts, and other Styles preferably may only use a subset of the available parts. This depends on the desired characteristics for a particular Style or SubStyle. For example, a 'cool' Style may not use a bridge part. Additionally, certain Styles that have a generally faster tempo preferably can use a virtually-doubled part size by simply doubling each part (i.e., intro, theme, theme, chorus, chorus, theme, theme, chorus, chorus, etc.).

Also, in certain cases, the user experience preferably may benefit from having the display updated for a particular Part. For example, an indication of the current position within the overall length of the song may be helpful to a user. Another example is to alert the user during the ending part that the song is about to end. Such an alert preferably might involve flashing a message (i.e., 'Ending') on some part of the display, and preferably will remind the user that they need to save the song now if they want it saved.

Another optimization at this level is preferably to allow changes made by the user during the interactive generation of a song to be saved on a part-by-part basis. This would allow the user to make a change to an instrument type, effect, volume, or filter, etc., and have that revised characteristic preferably be used every time that part is used. As an example, this would mean that once a user made some change(s) to a chorus, every subsequent occurrence of the chorus would contain that modified characteristic. Following this particular example, the other parts of the song would contain a default characteristic. Alternatively, the characteristic modifications preferably could either be applied to multiple parts, or preferably be saved in real time throughout the length of the song, as discussed further below.

Each Part preferably can be a different length, and preferably can be comprised of a series of SubParts. One aspect of a preferred embodiment involves the SubPart level disclosed in Fig. 15, but the use of the SubPart level is optional, in that the Part structure can be comprised directly by Sequences without the intervening SubPart level.

In certain embodiments, where a SubPart layer is implemented, each SubPart preferably can be of a different size. Such an approach can enhance the feel of the resulting musical composition, as it affords a degree of variety to the Parts.

Each SubPart preferably is comprised of a series of Sequences (SEQs). In keeping with the previous comment regarding the relationship between consistent sizing and flexibility of rule applications, each SEQ preferably can be the same length and time signature. In the example of Fig. 15, each SEQ is two bars long with a 4/4 time signature. Of course, these can be adjusted in certain variations of the invention, but in this example, this arrangement works well, because it allows us to illustrate how we can hold notes across a measure boundary. Typically, it might be advantageous to lengthen the size of the SEQs (as well as the RPs to be discussed hereinafter) to

allow greater diversity in the musical outcome. Such a variation is certainly within the scope of the present discussion, as well as Fig. 15.

Following the example of Fig. 15, each SEQ preferably consist of multiple Real Patterns (RPs) in parallel. Generally, it is useful to have 1 RP for each type of instrument. In this case, a type of instrument preferably corresponds to a single lane of the I-Way user interface (i.e., drums, bass, riff, etc.). RP data preferably is actual note data; generally, information at this level preferably would not be transposed unless through user interaction, and even then such interaction preferably would likely apply to multiple instruments. Of course this is a user interface decision, and is not a limitation to the embodiments discussed here.

In this case, the multiple RPs preferably are merged together to comprise the SEQ. As will be recognized by those skilled in the art, this is analogous to the way a state-of-the-art MIDI sequencer merges multiple sets of MIDI Type 1 information into MIDI Type 0 file.

Further background detail on this can be found in the "General MIDI Level 2 Specification" (available from the MIDI Manufacturer's Association) which is hereby incorporated by reference.

One reason for allowing multiple RPs in parallel to define a SEQ, is that at certain times, certain lanes on the I-Way may benefit from the use of multiple RPs. This is because it may be desirable to vary the characteristics of a particular piece of the music at different times during a song. For example, the lead preferably may be different during the chorus and the solo. In this case it may be desirable to vary the instrument type, group, filtering, reverb, volume, etc., and such variations can be enacted through the use of multiple RPs. Additionally, this method can be used to add/remove instruments in the course of play. Of course, this is not the only way to implement such variations, and it is not the only use for multiple RPs.

Following the example of Fig. 15, each RP preferably is comprised of two bars, labeled RPx and RPy. Such a two bar structure is useful because it preferably allows some variations in MIDI information (chord changes, sustain, etc.) across the internal bar boundary. Such variation can provide the effect of musical variation without adding the complexity of having chordal changes occur inside a bar, or having notes sustained among multiple RPs.

Generally, it is cumbersome to allow notes to be held over multiple RPs. This is partly because of the characteristics of MIDI, in that to hold a note you need to mask out the Note Off

command at the end of a pattern, and then mask out the Note On command at the beginning of the next pattern. Also, maintaining the same note across pattern boundaries is a concern when you switch chords, because the end of a pattern preferably is an opportunity to cycle through the chord progression, and you need to make sure that the old note being sustained is compatible with the new chord. The generation and merging of chord progression information preferably occurs in parallel with the activities of the present discussion, and shall be discussed below in more detail. While is considered undesirable to hold notes across patterns, there are exceptions.

One example of a potentially useful time to have open notes across multiple patterns is during Techno Styles when a long MIDI event is filtered over several patterns, herein called a 'pad'. One way to handle this example, is to use a pad sequence indicator flag to check if the current SEQ is the beginning, in the middle, or the end of a pad. Then the MIDI events in the pad track can be modified accordingly so that there will be no MIDI Note Offs for a pad at the beginning, no MIDI Note Ons at the beginning of subsequent RPs, and the proper MIDI Note Offs at the end.

Continuing our discussion of Fig. 15, RPs preferably are comprised of Virtual Patterns (VPs) that have had musical rules applied to them. Musical rules are part of the generation and merging of chord progression information that will be discussed in more detail below. A VP can be generally thought of as the rhythm of a corresponding RP, along with some general pitch information. Preferably, musical rules are applied to the VP, and the result is the RP. Musical rules are discussed in more detail below.

A VP preferably can be considered as a series of Blocks. In the example of Fig. 15, each Block has two dimensions: Blockd and Blockfx, but this is but one possible variation. In this example, Blockd corresponds to the data of the block, and Blockfx corresponds to effects that are applied to the data (i.e., volume, filtering, etc.). In this example, the Blockd information can be thought of as individual rhythmic pattern information blocks selected from a variety of possible rhythmic blocks (certain desirable approaches to create such a variety of possible rhythmic blocks, and the corresponding selection thereof in creating a VP, is discussed in greater detail later in this disclosure, with reference to Figs. 22 and 23).

The Blockfx dimension described in Fig. 15 is an optional way to add certain preferably characteristics to the Blockd information. For example, in addition to volume or filtering

information mentioned above, the Blockd dimension preferably can be used for allocation or distribution of musical information predictors, discussed in more detail below as Virtual Note/Controller (VNC) information. However, the Blockfx dimension is optional, and the Blockd information can be processed independently of such volume or filtering information, to great success.

Assuming the example presented earlier wherein the time signature is 4/4 and the RP is two bars, all Blocks in a pattern preferably must add up to 8 quarter notes in duration. In this example, assuming n Blocks in a particular RP, the duration in quarter notes of each Block in the corresponding VP would be between 1 and (8-{n-1}). While this example describes 4/4 time with a quarter note being the basic unit of length for a Block, simple variations to this example preferably would include alternate time signatures, and alternate basic units for the Block (i.e., 13/16 time signature and 32nd note, respectively, etc.).

Getting at the bottom of Fig. 15 we see an optional implementation of SubBlocks (SBs). Such an implementation could preferably be used, for example, for the drum lane of the I-Way during certain Styles, where it might be desirable to have separate SBs for the bass drum, cymbal, snare, etc. A further optimization of this implementation of the present embodiment would be to have the SB level of the drum lane preferably comprise directly the VP of the drum lane. Such an arrangement preferably would effectively remove the complexity of having a separate Blockfx for each individual SB of the drum lane. An example of where such an optimization might be useful when implementing the present invention is in an environment with limited resources, or an environment where having separate effects for separate parts of the drums (snare, bass drum, etc.) is not otherwise desirable.

Additionally, in some applications of the present invention, it may be desirable to enable certain levels in Fig. 15 to be bypassed. In such cases, this would preferably allow a user to input real pattern data in the form of actual note events (e.g., in real time during a song via a MIDI instrument as an input). Further, with the use of a companion PC software application (and a connection to the PC), in certain embodiments it is preferable to allow users to input their own MIDI patterns for use as Block data.

Various examples of preferred embodiments of the Music Rules used in the creation of a Song of the present invention will now be described.

Fig. 16 is a flow diagram depicting a general overview of a preferred approach to generating music in the context of the present invention. Starting at step 1, a style of music and a selected instrument are defined or loaded. Once the style of music and the type of instrument are known, the algorithm can apply Block rules to develop individual virtual pattern sub-blocks (e.g., those shown in Fig. 22). In certain alternative embodiments, the individual virtual pattern sub-blocks preferably are selected from a list or other data structure. Once the sub-blocks are available (e.g., from a list or from a block rule algorithm) they are processed into a Virtual Pattern (VP) at step 2. At this point in this example, a VP preferably is not music, although it does contain rhythmic information, and certain other embedded musical characteristics. At step 3, using the embedded musical characteristics of the VP data structure, musical rules preferably are applied to the VP to add more musicality to the pattern, and the result preferably contains both the rhythmic information of the VP, as well as actual musical information. At step 4 a tonic is preferably applied to the output from step 3, in that each measure preferably is musically transposed according to a tonic algorithm to impart a chordal progression to the data structures. Then at step 5, a mode preferably is applied that makes subtle changes to the musical information to output music information preferably set to a particular musical mode. Then, at step 6, a key preferably is applied to the data structure to allow key changes, and/or key consistency among various song components. Finally, at step 7, a global pitch adjustment preferably can be applied to the data structure, along with the rest of the song components, to allow real time pitch/tempo shifting during song play.

This process of applying various musical rules to generate a RP preferably can be a part of the overall song generation process mentioned above in connection with Fig. 15. Before going through the steps described in Fig. 16 in more detail, a discussion of the embedded characteristics mentioned above, as well as some mention of tonic and key theory will be helpful.

Bearing in mind that the MIDI Specification offers a concise way to digitally represent music, and that one significant destination of the output data from the presently discussed musical rules is the MIDI digital signal processor, we have found it advantageous to use a data format that has some similarities with the MIDI language. In the discussion that follows, we go through the steps of Fig. 16 in detail, with some examples of the data that can be used at each step. While the described data format is similar to MIDI, it is important to understand the

differences. Basically, the present discussion describes how we embed additional context-specific meaning in an otherwise MIDI compliant data stream. During processing at each of the steps in Fig. 16, elements of this embedded meaning preferably is extracted, and the stream preferably is modified in some musical way accordingly. Thus, one way to consider this process is that at each step, our stream becomes closer to the actual MIDI stream that is played by the MIDI DSP (this aspect is addressed in more detail below with reference to Fig. 21).

In the present example it is considered advantageous to break down the rhythmic and musical information involved in the music into Virtual Notes and/or Controllers (VNC). In the example of Fig. 17, we have provided several examples of VNCs that we have found to be useful. Basically, these VNCs represent our way of breaking down the musical rules of a particular genre into simplified mechanisms that can be used by an algorithm preferably along with a certain random aspect to generate new music that mimic the characteristics and variety of other original music in the genre. Depending on the Style of music, different types of VNCs will be useful. The list in Fig. 17 is simply to provide a few examples that will be discussed later in more detail.

In an important feature of this aspect of the present invention is that we have embedded control information for the music generation algorithm into the basic blocks of rhythmic data drawn upon by the algorithm. We have done this in a preferably very efficient manner that allows variety, upgradeability, and complexity in both the algorithm and the final musical output. A key aspect of this is that we preferably use a MIDI-type format to represent the basic blocks of rhythmic data, thus enabling duration, volume, timing, etc. Furthermore, we preferably can use the otherwise moot portions of the MIDI-type format of these basic blocks to embed the VNC data that informs the algorithm how to go about creating a part of the music. As an example, we preferably can use the pitch of each MIDI-type event in these basic sub-blocks of rhythmic data to indicate to the algorithm what VNC to invoke in association with that MIDI-type event. Thus, as this rhythmic data is accessed by the algorithm, the pitch-type data preferably is recognized as a particular VNC, and replaced by actual pitch information corresponding to the VNC function. Fig. 17 shows, in the first column, examples of such embedded values, and in the second and third columns, examples of recognized VNC nomenclature, and potential pitch information associated therewith.

In the example of Fig. 17, the fundamental type of VNC preferably is the Base Note. This can be considered in certain musical styles as the cornerstone of the melody, except, for example, when these notes are relatively short notes in a run. This is why rhythm exists in a VP to provide context to the VNCs. Example values of the Base Note are C,E,G or B. Which value is finally used preferably depends on a pseudo-random seed as part of an algorithm. We find that in these examples, these values provide pretty good music for the genres we have studied so far. The Magic Notes preferably can have the values indicated in Fig. 17 (assuming a diatonic scale is used), and these values are preferably relative to the preceding Base Note. Unlike a Base Note, Magic Notes preferably are useful at providing a note that does not strongly impact the melody. For example, the algorithm will see that the next note to be generated is a Magic Note 1, and it will use the Pseudo Random Number Seed to predictably select one of the possible values: +1, -1, +2, -2. The predictably-selected value preferably will be used to mathematically adjust the value from the preceding Base Note to preferably result in a note value. Following this example, if the preceding Base Note was a C2, and the result of the algorithm is to select a +1, then the Magic Note value is a D2. Note that preferably the only difference between Magic Note 0 and 1 is that Magic Note 0 can have a value of 0. Thus, the use of Magic Note 0 will occasionally result in a note that is the same value as the preceding Base Note. This is an example of a way to influence the sound of a particular Style in relatively subtle ways.

In the discussion above, by 'predictably-selected' we refer to the process of pseudo-randomly selecting a result based on a seed value. If the seed value is the same, then the result preferably will be the same. This is one way (though not the only way) to enable reproducibility. Further discussion of these pseudo random and seed issues is provided elsewhere in the present specification.

Continuing with Fig. 17, a High Note preferably simply adds an octave to the preceding Base Note, and is useful to make a big change in the melody. What is interesting here is that multiple VNCs preferably can occur in between the previous Base Note and the High Note, and this is a way to allow a musical phrase run to a tonic note, corresponding to an earlier Base Note. Obviously, this VNC is very useful, as it again preferably enables the structure of music to exist before the actual music itself is written. The algorithm preferably does not know what the final key, or mode will be at this point, but the octave and tonic preferably are available.

Similar to the Magic Note, the Harmonic Note VNC preferably allows the algorithm to pseudo-randomly select a harmonic from a set of possible harmonics. This capability is useful when there are multiple notes sounding at the same time in a chord. When this VNC is used, it preferably can result in any of the relative harmonics described in Fig. 17. These values are only examples of possible values, and ones that we find particularly useful for the types of music we have addressed.

Last Note is a VNC that is very similar to the Base Note, except that it preferably only contains a subset of the possible values. This is because, as we understand musical phrasing for the types of music we address, the final note preferably is particularly important, and generally sounds best when it has a relative value of C or G (bearing in mind that in this example, all the notes preferably can subsequently be transposed up or down through additional steps). As with all the VNCs, the precise note that might be played for this value preferably depends on the Mode and Key applied subsequently, as well as general pitch shifting available to the user. However, in the music we address, we find this to be a useful way to add subtlety to the music, that provides a variety of possible outcomes.

One Before Last Note is a VNC that preferably immediately precedes the Last Note. Again, this is because we have found that the last two notes, and the harmonic interval between them, are important to the final effect of a piece, and accordingly, we find it advantageous with the Final Notes of C and G to use One Before Last Notes of E, G, or B. These values can be adapted for other Styles of music, and only represent an example of how the VNC structure can be effectively utilized.

The last example VNC in Fig. 17 is the ALC controller. This is one example of how certain musical non-pitch concepts can preferably be employed using a MIDI controller. In this example, the ALC controller can be thought of as a prefix which modifies the meaning of immediately following notes. The ALC controller can be used to indicate that the next note is to be treated in a special manner, for example, to setup a chord. In this example, you can use a particular predefined value for the ALC controller to precede a sequence of a fixed note with additional harmonic notes. Similar to the Magic Note VNC discussed above, the Harmonic Notes following a ALC controller preferably allow the algorithm to pseudo-randomly select a harmonic from a set of possible harmonics. This capability is useful when there are multiple notes

sounding at the same time in a chord. When this VNC is used, it preferably can result in any of the relative harmonics described in Fig. 17. These values are only examples of possible values, and ones that have been found particularly useful for the types of music addressed up to the time hereof. Another example use of the ALC controller is to setup fixed notes. In this case, preferably one follows the appropriate ALC controller with Fixed Note values for any desired actual note value. This approach is useful in many instances to have a more carefully limited song output where a particular interval between notes in the desired music can be achieved. Additionally, playing well-known phrases or sequences preferably is possible with this use of the ALC controller. One preferably could encode portions of an entire song this way to have a piece that closely resembles an existing musical piece. In this example, one preferably could have certain parts of the music still interactively generated to enable a song to sound just like an existing song (in melody, for example), yet preferably still allow other parts to be different (like bass or drums, for example).

In this manner, you can setup the resulting chord because the ALC value preferably will alert the software routine that is processing all of the VNCs to let it know that the following note is to be the basis of a chord, and that the next number of harmonic notes will be played at the same as the basis note, resulting in a chord being played at once. This example shows one way that this can be done effectively. Other values of VNC controllers preferably can be used to perform similar musical functions.

It is important to note that an additional variation can preferably be implemented that addresses the natural range, or Tessitura, of a particular instrument type. While the software algorithm preferably is taking the VNCs mentioned above and selecting real values, the real pitch value preferably can be compared to the real natural range of the instrument type, and the value of subsequent VNC outcomes preferably can be inverted accordingly. For example, if the Base Note of a given pattern is near the top of the range for a bass instrument Tessitura, any subsequent Magic Notes that end up returning a positive number can be inverted to shift the note to be below the preceding Base Note. This is a particular optimization that adds subtlety and depth to the outcome, as it preferably incorporates the natural range limitations of particular instrument types.

As a simplified example of Tessitura, Fig. 18 depicts the relative optimal ranges of particular instrument types. In the present context, the Tessitura of an instrument preferably is the range at which it sounds optimal. Certain sounds in the MIDI sound bank preferably are optimized for particular ranges. If you select a bass guitar sound and play very high pitched notes, the result may not be very good. For higher pitches, a guitar or violin sound may work better. Accordingly, when the musical rule algorithm is processing VNCs , the Tessitura of the selected instrument type preferably can play a role in the outcome of the real note value generated. If the selected instrument is approaching the top edge of its' Tessitura, and the musical rule routine comes across a High Note VNC, then the algorithm preferably can be designed to bump the generated pitch down an octave or two. Similarly, other VNCs can be processed with deference to the Tessitura of the selected instrument.

Fig. 19 describes another aspect of this musical process. Musical Key changes preferably can be encoded as offsets. By this we mean that given a Key of X, the Key can be shifted up or down by inserting an offset. Such an offset preferably will transpose everything by the exact value to result in a musical phrase that is exactly as it was, but now in a different Key. Fig. 19 has as examples the Keys of A, C, D, and G. A Key of C preferably would have an offset of 0, A an offset of –3, D an offset of +2, and G an offset of +8. As will be appreciated by a student of Musical Theory, the offset preferably corresponds closely with a number of half steps in an interval. The interval between C and G is 8 half steps. Other Keys can be similarly achieved.

The use of halfsteps for encoding Keys is advantageous because, as mentioned previously, the MIDI language format uses whole numbers to delineate musical pitches, with each whole number value incrementally corresponding to a half step pitch value. Other means of providing an offset value to indicate Keys can be applied, but in our experience, the use of half steps is particularly useful in this implementation because of we are preferably using a MIDI DSP, and so the output of the Musical Rules preferably will be at least partly MIDI based.

Fig. 20 describes another Musical Rule that preferably is part of the overall process: Mode application. As can be appreciated by a student of Musical Theory, assuming the mode is described in terms of sharps (as opposed to flats) the particular placement of sharps is a large part of what gives each musical phrase its own identity. In Fig. 20 we give the example of a Lydian Mode, with Ascending or Descending versions preferably available. Other well established

musical modes exist (Ionian, Dorian, Hypodorian, Phrygian, Hypophrygian, Hypolydian, Mixolydian, Aeolian, Locrian, etc.) and we only use Lydian here in the interests of space. Clearly, the present invention can involve other modes, with corresponding values as those in Fig. 20. In cases where a mode is desired that is not a conventional western mode, it is preferable to upgrade or alter the soundbank (e.g., located in Flash 49) so that other musical intervals are possible.

Fig. 20 begins with a list of all preferably available notes in the genre of music that we are addressing. That is followed by the corresponding preferably natural note values that we term Natural Mode. The values of notes in the Natural Mode preferably correspond to the All Notes row of notes without the sharps (again assuming that in the present discussion we are defining our modes in terms of sharps, and not flats). Then the Lydian mode preferably is listed, which does not allow F naturals. In order to decide whether an F natural is to be raised to the next available pitch of F sharp, or lowered to the next available pitch of E, an algorithm preferably will decide between an ascending or descending transposition. Accordingly, a descendingly transposed F natural preferably will be changed to an E, and an ascendingly transposed F natural preferably will be transposed to a F sharp. Given that sharps vary from the Natural Mode, the use of an ascending Lydian Mode results in music that has more F sharps, and is thus more aggressively Lydian. This general concept is evident in other Modes as well, with ascending transpositions typically being more aggressive than descending transpositions.

At this point we will go through a detailed example of the Musical Rule portion of the algorithm, using Fig. 21 as the example. This discussion will incorporate the earlier discussions of the preceding figures, to demonstrate how a preferred embodiment of the present invention preferably incorporates them.

Fig. 21 depicts the data as it preferably exists between each of the numbered steps 2 – 6 in Fig. 16. The Musical Notation is represented to clarify the overall concept, as well as to indicate a simplified example of the preferable format the data can take in the software routine.

Beginning at the top row, there is a collection of predefined VP Sub-Blocks that preferably can advantageously be indexed by music Style and/or length. These blocks preferably are of variable sizes and preferably are stored in a hexadecimal format corresponding to the notation of pitch (recognizing that in certain embodiments the pitch information of a VP does not

represent actual pitch characteristics, but VNC data as discussed above), velocity, and duration of a MIDI file (the preferable collection of predefined VP-Sub-Blocks is discussed in more detail below with reference to Figs. 22 – 23). As shown in the top row of Fig. 21, Rests preferably are also available in this collection of available patterns. This collection of indexed Sub-Blocks preferably is used by a software routine to construct Virtual Patterns (VPs). As mentioned earlier, certain alternative embodiments preferably involve using algorithmic block rules to generate the collection of Sub-Blocks. Such algorithmic rules preferably are configured to accept the music style and instrument type as inputs to then output a collection of Sub-Blocks that are appropriate for that style/instrument combination. Whether the Sub-Blocks are selected from predefined collection, or generated on the fly with an algorithm, they preferably are organized into a VP. VPs preferably are a collection of Sub-Blocks that have been assembled by the routine into preferably consistently-sized groupings.

After step 2 of Fig. 16 is applied, we preferably have a VP. The second row of Fig. 21 (VP) depicts an example VP that is 2 bars long, and composed of the following sequence: Base Note, Magic Note 1, Magic Note 0, High Note, and another Base Note. Note that at this time the rhythm of the part preferably is in place, and the value of each note is conceptually the embedded VNC information. If the VP is played at this point, the output would likely not be pleasing. The right column of row 2 depicts the format that this data preferably is stored in; as is discussed elsewhere in this disclosure, this format is remarkable similar to MIDI format data, with one exception being that the VNC information preferably is implicitly embedded in the data stream.

The third row (NCP) depicts the same data after step 3 of Fig. 16 is applied. The VNCs embedded in the VP from row 2 preferably have been interpreted by the routine with the help of pseudo-random selections from the possible VNC values. Thus, for the first Base Note in row 2, we have a real note value of E in row 3, and for the Magic Note Type 1 of row 2 we have decremented the previous Base Note two half steps to a D in row 3. For the Magic Note Type 0 we have adjusted the previous value by 0, resulting in another D. This goes on through the VP, and the result is clear in row 3. At this point, we preferably have the basic musical information that will end up in the song, except that the Chord and Mode transpositions preferably have not yet been made.

The fourth row in Fig. 21 (PwT) depicts the data stream after step 4 of Fig. 16 is applied. As can be seen, the NCP of row 3 has been transposed down. This is to allow the particular pattern being constructed to preferably conform to a particular Tonic note, thus placing it into a suitable chord preferably to match the other elements of the musical piece. This feature allows different portions of the melody preferably to conform to different tonic notes, thus preferably proceeding through a chord progression, while ensuring that all instruments preferably conform to the same chord progression.

Row 5 of Fig. 21 (PwTM) takes the pattern of notes and preferably conforms it to a particular Mode (e.g., Ionian, Dorian, Hypodorian, Phrygian, Hypophrygian, Lydian, Hypolydian, Mixolydian, Aeolian, Locrian, etc.) preferably as well as a particular Mode type (like descending, ascending, etc.). A more complete list of musical modes and mode types has been prepared by Manuel Op de Coul (available on the world wide web at: www.xs4all.nl/~huygensf/doc/modename.html) and is hereby incorporated herein by reference. The conformation of the pattern of notes to a particular Mode preferably is done in a manner consistent with Fig. 20, discussed above. In the example of Fig. 21, the resulting musical phrase is very similar to that of Row 4, except the notable difference of the C sharp being reduced to a C. This is because there is no such C sharp in the Lydian mode, and so it's removal is preferably required at this step. If the Modal adjustment were using the Lydian ascending mode, which is more aggressively ascending because there are more sharps, this C sharp would have preferably 'rounded up' to the next Lydian note of D. But, since in this example we are using a Lydian descending mode, the C sharp is preferably 'rounded-down' to a C.

The final row of Fig. 21 (RP) indicates the point when the musical phrase preferably can be globally transposed up or down the scale. This is advantageous in the case where a global pitch adjustment feature is desired to preferably allow the user to quickly and easily shift the pitch of a song up or down (such as is discussed in an earlier example of the Pitch/Tempo key used in combination with the Up/Down keys). The example of Row 6 shows a transposition of 2 half steps. As with all the rows of this figure, this can be seen in the musical notation, as well as the software notation, where the third pair of numbers can be seen to increment by a value of two, for each line.

There are instances where certain elements of the music preferably do not need the musical rules discussed above to be invoked. For example, drum tracks preferably do not typically relate to Mode or Key, and thus preferably do not need to be transposed. Additionally, many instrument types such as drums, and MIDI effects, preferably are not arranged in the MIDI sound bank in a series of pitches, but in a series of sounds that may or may not resemble each other. In the example of drums, the sound corresponding to C sharp may be a snare drum sound, and C may be a bass drum sound. This means that in certain cases, different levels of the process discussed above in reference to Fig. 21 preferably may be advantageously bypassed in these cases.

The collection of sub-blocks discussed above, from which VPs preferably are constructed, can be better understood in light of Figs. 22 and 23.

Fig. 22 depicts an example of the rhythmic variations that preferably are possible, based on example durations of 1 or 2 quarter notes. The first row indicates the 4 possible variations, given a few basic conditions: that the eighth note is the smallest unit, the length is 1 quarter note, and that all full rests are indicated separately as 'empty'. The second row in Fig. 22 lists the possible variations, given similar variations: that the eighth note is the smallest unit, that any variations in the first row are not included, and that the length is 2 quarter notes.

One way to create a set of rhythmic variations such as those in Fig. 22 preferably is to put the variation data into MIDI event format. This approach preferably involves using a MIDI sequencer software tool (such as Sonar from Cakewalk, and Cubase from Steinberg) to generate the rhythmic blocks. This preferably allows the use of a variety of input methods (e.g., a keyboard controller, a MIDI wind controller, a MIDI guitar controller, etc.), and further preferably allows the intuitive copying, pasting, quantizing, and global characteristic adjustments (e.g., selecting multiple events and adjusting the pitch for all). Then, the MIDI events preferably can be exported as a MIDI file (possibly 1 file for each instrument group). Finally, a software batch file program preferably can be written to open the MIDI file and parse out the substantial header information, as well as any unneeded characteristic information (such as controller or patch information), and preferably output the optimized data into a file that is suitable to include in the source code (e.g., ASCII text tables). The use of the sequencing tool preferably enables one to quickly generate a variety of appropriate rhythmic blocks for a given instrument type,

since the vast array of MIDI controller devices are available that can mimic the characteristics of a particular instrument type. For example, one can use a MIDI guitar controller to strum in patterns for a guitar type of instrument group.

The example of Fig. 22 is simplified to convey a concept; that all rhythmic variations covering up to two quarter notes (given the conditions discussed above) preferably can be organized very efficiently according to rhythmic density. Fig. 22 teaches an advantageous way to efficiently organize the set of blocks used to construct a VP shown in Fig. 15. If the example of Fig. 22 were expanded to include additional rows for rhythmic blocks with longer durations, given conditions such as those described above that are consistent across the rows, then each subsequent row would have patterns of less density than those above it. This is because of the condition that each row does not include any of the variations present in rows above it, and because the duration of the pattern increases for each subsequent row. Thus, there is a direct relationship between the example shown in Fig. 22 and the relative rhythmic density of patterns used to make a VP.

Clearly, if any of the conditions described in Fig. 22 were changed, e.g., if a sixteenth note were the smallest unit or full rests were indicated with a pattern containing a rest, then preferably the number of variations would be different. While the number would be different, the desirable effects of organizing patterns based on this concept of rhythmic density would remain.

In addition to efficiency, such an approach to organizing the available rhythmic blocks preferably enables the use of rhythmic density as an input to a software (e.g., algorithmic function) or hardware (e.g., state table gate array) routine. Thus, one preferably can associate a relative rhythmic density with a particular instrument type and use that rhythmic density, possibly in the form of a desired block length, preferably to obtain a corresponding rhythmic block. This preferably can be repeated until a VP is complete (see Fig. 15). The VP preferably can thereby be constructed with a desired relative rhythmic density. This is particularly useful because it preferably allows the creation of VPs with almost limitless variations that have rhythmic characteristics preferably generally corresponding to a given instrument type.

As will be apparent to one of ordinary skill in the art of MIDI, given the context of VP generation discussed herein, the rhythmic variations shown in Fig. 22 can be represented in the

form of MIDI events. In this case, many of the available characteristics in the MIDI events, such as pitch, velocity, aftertouch, etc., preferably might be generically set. Then, additional functions for such characteristics preferably can be applied to the MIDI events during the creation of VPs to impart additional subtlety to the finished music. Such functions preferably can be fairly simple and still be effective. As one example, for a given Style of music (e.g., rock), the velocity of any MIDI events in the VP that fall on a particular location in the measure (e.g., the downbeat) can be modestly increased. Similarly, in a music Style that generally has a rhythmic swing feel, where one or more of the beats in a measure may be slightly retarded or advances, the corresponding MIDI events in a VP preferably can be modified so as to slightly adjust the timing information. Clearly, these types of simple functions preferably can be selectively applied to either a given instrument type, and/or a given musical Style.

Similar to the concept of using relative rhythmic density as a deterministic characteristic in creating algorithmic music, Fig. 23 describes a concept of relative mobility of note pitch. As shown in Fig. 23, the vertical axis indicates pitch change, and the horizontal axis indicates time. Two example types of melody streams are depicted; the top having a fluid movement through a variety of pitches, and the bottom having rather abrupt, discrete changes among a fewer number of pitches. Thus, the melody on the top of Fig. 23 has a higher relative mobility of note pitch. As can be appreciated by the previous discussion of VNCs, the melody example on the top preferably would generally require more Magic Notes to imitate, and the melody example on the bottom preferably would generally require more Base Notes and High Notes to imitate.

This concept preferably applies to most instrument types in a given musical Style as well, in that certain instruments have a higher relative mobility of note pitch than others. As an example, a bass guitar in a rock Style can be thought of as having a lower relative mobility of note pitch compared to a guitar in the same Style. The relationship between relative mobility of note pitch and relevant VNC type can be very helpful in creating the collection of predefined sub-blocks discussed above, in that it serves as a guide in the determination of actual VNC for each rhythmic pattern. When one wants to create a set of rhythmic building blocks for use in a particular musical Style and/or instrument type, it is advantageous to consider/determine the desired relative mobility of note pitch, and allocate VNC types accordingly.

As an additional variation, and in keeping with the discussion above regarding relative rhythmic density, an architecture that constructs a VP for a given instrument type and/or musical Style preferably can greatly benefit from a software (e.g., algorithmic function) or hardware (e.g., state table gate array) routine relating to relative mobility of note pitch. As an example, a particular music Style and/or instrument type can be assigned a relative rhythmic density value, and such a value can be used to influence the allocation or distribution of VNC types during the generation of a VP.

The use of relative rhythmic density and relative mobility of note pitch in the present context preferably provides a way to generate VPs that closely mimic the aesthetic subtleties of 'real' human-generated music. This is because it is a way of preferably quantifying certain aspects of the musical components of such 'real' music so that it preferably can be mimicked with a computer system, as disclosed herein. Another variation and benefit of such an approach is that these characteristics preferably are easily quantified as parameters that can be changeable by the user. Thus a given musical Style, and/or a given instrument type, preferably can have a relative mobility of note pitch parameter (and/or a relative rhythmic density parameter) as a changeable characteristic. Accordingly, the user preferably could adjust such a parameter during the song playback/generation and have another level of control over the musical outcome.

Various examples of preferred embodiments for the block creation aspects of the present invention will now be described.

Continuing the example presented in Fig. 15, wherein a RP preferably is 2 bars, and a VP preferably is comprised of 8 quarter notes (QN), the pattern structure creation example of Fig. 24 assumes that the particular song generation implementation preferably involves a VP length of 8 QN, a 2 bar RP, and variably-sized Blocks. While those skilled in the art will appreciate the considerable number of advantages arising from the architecture of this preferred embodiment, they will additionally appreciate that various adaptations and modifications to these embodiments can be configured without departing from the spirit and scope of the invention.

As shown in Fig. 24, one preferred embodiment of the present invention involves the creation of a pattern structure. This pattern structure preferably is comprised of the information needed to select the actual Blocks, which in many ways are the fundamental unit of the song generation. This example of pattern structure creation involves determining each Block's

duration (in a given VP), as well as the group of instruments from which the Block will be selected. Following this step, and discussed below, this information preferably is used to directly generate the Blocks themselves.

Patt_Info is a routine that preferably can be used to generate the pattern structure information as part of the creation of a particular VP from Blocks.

Shift is a multiplier that preferably can be used in a variety of ways to add variation to the composed VP; for example, it could be a binary state that allows different Block variations based on which of the 2 bars in the RP that a particular Block is in. Other uses of a Shift multiplier can easily be applied that would provide similar variety to the overall song structure.

Num_Types is the number of instruments, and Num_Sub_Drums is the number of individual drums that make up the drum instrument. This latter point is a preferable variation that allows an enhanced layer of instrument selection, and it can be applied to other contexts other than the drum instrument. Conversely, this variation is not at all necessary to the present invention, or even the present embodiment.

Block_Ind is the Block index, FX_No is for any effects number information. Combi_No is an index that preferably points to a location in a table called Comb_Index_List. This table preferably is the size of the number of Styles multiplied by the number of instrument types; each entry preferably contains: SubStyle_Mask to determine if the particular entry is suitable for the present SubStyle, Combi_Index to determine the Block length, and Group_Index to determine the group of individual MIDI patches (and related information) from which to determine the Block.

Combi_Index preferably points to a table called Style_Type_Combi that preferably contains multiple sets of Block sizes. Each Block_Size preferably is a set of Block sizes that add up to the length of the SEQ. An example SEQ length is 8 QN.

Group_Index preferably points to a table called Style_Group that preferably contains sets of MIDI-type information for each group of Styles, preferably organized by MIDI Bank. PC refers to Patch Change MIDI information, P refers to variably sized MIDI parameters for a given Patch, and GS stands for Group Size. GS for group 1 preferably would indicate how many instruments are defined for group 1.

One preferable optimization of the execution of this step is to incorporate a pseudo-random number generator (PRNG) that preferably will select a particular patch configuration from the group identified by GS. Then, as the user elects to change the instrument within a particular SubStyle, and within a particular lane, another set of patch information preferably is selected from the group identified by GS. This use of a PRNG preferably can also be incorporated in the auto-generation of a song, where, at different times, the instrument preferably can be changed to provide variation or other characteristics to a given song, Part, SubPart, SEQ, RP, VP, etc. There are other areas in this routine process that preferably could benefit from the use of a PRNG function, as will be obvious to one of ordinary skill in the art.

Once the Block duration and instrument patch information preferably are determined for a given VP, the virtual Block information preferably can be determined on a Block-by-Block basis, as shown in Fig. 25.

Block_List preferably is a routine that can determine a virtual Block using the Block size, and the instrument type. As shown in Fig. 25, Style preferably is a pointer to a table of Virtual_Block_Data pointers that preferably are organized by Width (i.e., 1-8 QN) and Group (i.e., instrument group). Once the Start_Pointer is determined, the Block data preferably can be obtained from a Virtual_Block_Data table. Special cases exist where the Block data may be already known; for example, empty Blocks, repeating Blocks, etc.

Again, as discussed above in connection with the pattern structure generation, the present steps of the overall process preferably can use an optional PRNG routine to provide additional variety to the Block. Another fairly straightforward extension of this example is to use 'stuffing' (i.e.; duplicate entries in a particular table) preferably to provide a simple means of weighting the result. By this we refer to the ability to influence the particular Block data that is selected from the Virtual_Block_Data table preferably by inserting various duplicate entries. This concept of stuffing can easily be applied to other tables discussed elsewhere in this specification, and other means of weighting the results for each table lookup that are commonly known in the art can be easily applied here without departing from the spirit and scope of the invention.

Additionally, as one of ordinary skill in the art will appreciate, though these examples of preferred embodiments to the various inventive steps involve substantial reliance on tables, it would be fairly easy to apply concepts of state machines, commonly known in the art, to these

steps and optimize the table architecture into one that incorporates state machines. Such an optimization would not depart from the spirit and scope of the present invention.

Various examples of preferred embodiments for pseudo-random number generation aspects of the present invention will now be described.

Some of the embodiments discussed in the present disclosure preferably involve maximizing the limited resources of a small, portable architecture, preferably to obtain a complex music generation/interaction device. When possible, in such embodiments (and others), preferably it is desirable to minimize the number of separate PRNG routines. Although an application like music generation/interaction preferably relies heavily on PRNG techniques to obtain a sense of realism paralleling that of similarly Styled, human-composed music, it is tremendously desirable to minimize the code overhead in the end product so as to allow the technology preferably to be portable, and to minimize the costs associated with the design and manufacture. Consequently, we have competing goals of minimal PRNG code/routines, and maximal random influence on part generation.

In addition, another goal of the present technology is preferably to allow a user to save a song in an efficient way. Rather than storing a song as an audio stream (i.e.; MP3, WMA, WAV, etc.), it is highly desirable to save the configuration information that was used to generate the song, so that it preferably can be re-generated in a manner flawlessly consistent with the original. The desirability of this goal can easily be understood, as a 5 minute MP3 file is approximately 5MB, and the corresponding file size for an identical song, preferably using the present architecture, is approximately 0.5KB, thus preferably reduced by a factor of approximately 10,000. In certain preferred embodiments, the sound quality of a saved song is similar to a conventional compact disc (thereby demonstrably better than MP3). In this comparison, a 5 minute song stored on a compact disc might be approximately 50MB; thus the file size of a song using the present invention is reduced from a compact disc file by a factor of approximately 100,000.

Saving the configuration information itself, rather than an audio stream, preferably allows the user to pick up where they left off, in that they can load a previously saved piece of music, and continue working with it. Such an advantage is not easily possible with a single, combined audio stream, and to divide the audio into multiple streams would exponentially increase the file

size, and would not be realizable in the current architecture without significant trade-offs in portability and/or quality.

Additionally this aspect of the present invention preferably enables the user to save an entire song from any point in the song. The user preferably can decide to save the song at the end of the song, after experiencing and interacting with the music creation. Such a feature is clearly advantageous as it affords greater flexibility and simplicity to the user in the music creation process.

Turning now to Fig. 26, we have a diagram representing the preferable algorithmic context for some examples of Pseudo-Random Number Generation (PRNG). Drum Seed (DS) is a number that preferably is used as input to a simple PRNG routine to generate DS0-DS4. As would be apparent to one of ordinary skill in this art, the number of outputs preferably can be varied; we use 4 here for illustrative purposes. The 4 values that are output from the PRNG preferably are fed into various parts of the Drum Part Generation Algorithm to provide some pseudo-random variation to the drum part.

It is important to note that if the same seed input to the simple PRNG routine is used a plurality of times, the same list of values preferably will be output each time. This is because simple PRNG routines are not random at all, as they are a part of a computing system that is, by its very nature, extremely repeatable and predictable. Even if one adds some levels of complexity to a PRNG algorithm that take advantage of seemingly unrelated things like clocks, etc., the end user can discern some level of predictability to the operation of the music generation. As can be imagined, this is highly undesirable, as one of the main aspects of the device is to generate large quantities of good music.

One benefit of the preferably predictable nature of simple PRNGs is that, by saving the seed values, one preferably can generate identical results later using the same algorithm. Given the same algorithm (or a compatible one, preferably), the seeds preferably can be provided as inputs and preferably achieve the exact same results every time. Further discussion of the use of seeds in the music generation/interaction process is discussed elsewhere in this specification.

While it is a feature of the present invention to preferably incorporate PRNG that are repeatable, there are also aspects of the present invention that preferably benefit from a more 'truly-random' number generation algorithm. For purposes of clarity, we call this 'complex

PRNG'. Using the example of Fig. 26 and 27, if, on a regular basis, the same seed input were used for both the Drum part and the Bass part, it might limit the variability of the outcome. Another example is that, although preferably when playing a previously saved song, you want A and A' to always be the same, when you are generating a new song, it preferably is highly desirable that these seed inputs be randomly different. Otherwise the song generation suffers from the same repeatability as the song playback.

One example of a complex PRNG that works within the cost/resource constraints we have set, is one preferably with an algorithm that incorporates the timing of an individual user's button-presses. For example, from time to time in the process of generating music and providing user interaction in that generative process, we preferably can initialize a simple timer, and wait for a user button press. Then the value of that timer preferably can be incorporated into the PRNG routine to add randomness. By way of example, one can see that, if the system is running at or around 33 MHz, the number of clocks between any given point and a user's button press is going to impart randomness to the PRNG. Another example is one preferably with an algorithm that keeps track of the elapsed time for the main software loop to complete; such a loop will take different amounts of time to complete virtually every time it completes one loop because it varies based on external events such as user button presses, music composition variations, each of which may call other routines and/or timing loops or the like for various events or actions, etc. While it preferably is not desirable to use such a complex PRNG in the generation of values from seeds, due to repeatability issues discussed above, it preferably can be desirable to use such a PRNG in the creation of seeds, etc., as discussed above. As an additional example, such a complex PRNG routine can be used to time interval, from the moment the unit is powered up, to the moment the 'press-it-and-forget-it' mode is invoked; providing a degree of randomness and variability to the selection of the first auto-play song in Home mode (discussed earlier in this disclosure). Of course, this type of complex PRNG preferably is a variation of the present invention, and is not required to practice the invention.

One desirable aspect of the present invention involves the limiting of choices to the end user. The various ways instruments can be played are limitless, and in the absence of a structure, many of the possible ways can be unpleasant to the ear. One feature of palatable music is that it conforms to some sort of structure. In fact, it can be argued that the definition of creativity is

expression through structure. Different types of music and/or instruments can have differing structures, but the structure itself is vital to the appeal of the music, as it provides a framework for the listener to interpret the music. The present invention involves several preferable aspects of using seed values in the generation of a piece of music. One preferable way to incorporate seeds is to use two categories of seeds in a song: 1) seeds determining/effecting the higher-level song structure, and 2) seeds determining/effecting the particular instrument parts and characteristics. Preferably, the first category of seeds is not user-changeable, but is determined/effected by the Style/SubStyle and Instrument Type selections. Preferably, the second category of seeds is user-changeable, and relates to specific patterns, melodies, effects, etc. The point in this example is that there are some aspects of the music generation that are preferably best kept away from the user. This variation allows the user to have direct access to a subset of the seeds that are used for the music generation, and can be thought to provide a structure for the user to express through. This preferable implementation of the present discussion of seeds enables a non-musically-trained end user to creatively make music that sounds pleasurable.

Various examples of preferred embodiments for a simple data structure (SDS) to store a song of the present invention will now be described.

The use of PRNG seeds preferably enables a simple and extremely efficient way to store a song. In one embodiment of the present invention, the song preferably is stored using the original set of seeds along with a small set of parameters. The small set of parameters preferably is for storing real time events and extraneous information external to the musical rules algorithms discussed above. PRNG seed values preferably are used as initial inputs for the musical rules algorithms, preferably in a manner consistent with the PRNG discussion above.

Fig. 28 lists some examples of the types of information in an SDS:

'Application Number' is preferably used to store the firmware/application version used to generate the data structure. This is particularly helpful in cases where the firmware is upgradeable, and the SDS may be shared to multiple users. Keeping track of the version of software used to create the SDS is preferable when building in compatibility across multiple generation/variations of software/firmware.

'Style/SubStyle' preferably is used to indicate the SubStyle of music. This is helpful when initializing various variables and routines, to preferably alert the system that the rules associated with a particular SubStyle will govern the song generation process.

'Sound Bank/Synth Type' preferably indicates the particular sound(s) that will be used in the song. This preferably can be a way to preload the sound settings for the Midi DSP.

'Sample Frequency' preferably is a setting that can be used to indicate how often samples will be played. Alternatively, this preferably can indicate the rate at which the sample is decoded; a technique useful for adjusting the frequency of sample playback.

'Sample set' preferably is for listing all the samples that are associated with the Style of music. Although these samples preferably may not all be used in the saved SDS version of the song, this list preferably allows a user to further select and play relevant samples during song playback.

'Key' preferably is used to indicate the first key used in the song. Preferably, one way to indicate this is with a pitch offset.

'Tempo' preferably is used to indicate the start tempo of the song. Preferably, one way to indicate this is with pulses per quarter note (PPQN) information.

'Instrument' preferably is data that identifies a particular instrument in a group of instruments. Such as an acoustic nylon string guitar among a group of all guitar sounds. This data is preferably indexed by instrument type.

'State' preferably is data that indicates the state of a particular instrument. Examples of states are: muted, un-muted, normal, Forced play, solo, etc.

'Parameter' preferably is data that indicates values for various instrument parameters, such as volume, pan, timbre, etc.

'PRNG Seed Values' preferably is a series of numerical values that are used to initialize the pseudo-random number generation (PRNG) routines. These values preferably represent a particularly efficient method for storing the song by taking advantage of the inherently predictable nature of PRNG to enable the recreation of the entire song. This aspect of the present invention is discussed in greater detail previously with respect to Figs. 26 and 27.

Through the use of these example parameters in a SDS, a user song preferably can be efficiently stored and shared. Though the specific parameter types preferably can be varied, the

use of such parameters, as well as the PRNG Seeds discussed elsewhere in this disclosure, preferably enables all the details necessary to accurately repeat a song from scratch. It is expected that the use of this type of arrangement will be advantageous in a variety of fields where music can be faithfully reproduced with a very efficient data structure.

Fig. 29 depicts a logical flow chart for a preferable general architecture that could be used in combination with the SDS to practice the present invention. This flow chart describes the big picture for a preferable software/firmware implementation, and describes in more detail how the song preferably is efficiently and interactively generated using seed values.

At the start of Fig. 29, an initial set of seed values preferably is either loaded from a data file (e.g., SDS) or determined anew (e.g., using the Complex PRNG approach discussed elsewhere in this disclosure). While this set of values preferably can effectively be determined/loaded for the entire song at this point, it may be considered advantageous to only determine/load them in sections as needed, preferably to provide a degree of randomness to a freshly generated song. Further, as discussed above, the seed values may preferably be arranged in two categories, one user-changeable, and the other not. Once at least some seed values preferably are determined/loaded, the music for a given song part preferably begins to be generated, and the user interface (e.g., display, video output, force-feedback, etc.) preferably can be updated accordingly. At any point in this process, if a user input is detected (other than a 'save' command), such as a change of instrument or effect, the relevant seeds for the part of the song currently being changed by the user preferably are updated and the generation of the music for the given part preferably continues. If a user input 'save' command is detected, all seeds (not just the relevant seeds for the given song part) preferably can be saved to a non-temporary storage location, such as Flash memory, a hard drive, or some other writeable memory storage location that affords some degree of permanence. This arrangement is desirable because it preferably allows a user to listen to most of a song before electing to save it in its entirety. As long as there is no user input, the generation of music for a given song part preferably continues until the end of song part is detected, at which time the flow preferably proceeds to the next song part. At this time, if necessary, the relevant seeds for the next song part preferably are determined/loaded. Eventually, when an end-of-song condition preferably is detected, the song ends.

Various examples of preferred embodiments for a complex data structure to store a song of the present invention will now be described.

In another variation to the present invention, it is contemplated that, for purposes of saving and playing back songs, the reliance on seeds as inputs to the musical rule algorithms (see SDS discussion above) preferably may be exchanged for the use of Complex Data Structures (CDS). In part because of it's efficiency, the seed-based architecture discussed above is desirable when forward/backward compatibility is not an issue. However, it has some aspects that may not be desirable, if compatibility across platforms and/or firmware revisions is desired. In these cases, the use of an alternative embodiment may be desirable.

As described above, a seed preferably is input to a simple PRNG and a series of values preferably are generated that are used in the song creation algorithm. For purposes of song save and playback, the repeatability preferably is vital. However, if the algorithm is modified in a subsequent version of firmware, or if other algorithms would benefit from the use of the simple PRNG, while it is in the middle of computing a series (e.g.; DS0-DS3 in Fig. 26), or if additional elements are needed for subsequent music Styles, etc., that involve additional seeds, it is possible that the repeatability and backwards-compatibility may be adversely impacted. This means that in certain applications of the present invention, preferably in order to allow future upgrades to have significant leeway, and in order to maintain backwards-compatibility with songs saved before the upgrade, another preferably more complex data structure for saving the song is desirable.

Fig. 30 describes some example parameters to include in such a CDS. In general, the difference between this structure and the SDS example described in Fig. 28 is that this preferably does not rely on seed values to recreate the song. Instead, this CDS preferably captures more of the actual data in the song, resulting in a file size that is larger than the SDS example. The use of CDS preferably is still a tremendously more efficient and desirable means of saving a song compared to an audio stream, as mentioned above in connection with the seed method. While the seed method preferably gives you a size reduction over a typical MP3 audio stream of 10,000, the CDS method preferably might give an approximate size reduction of 1,000; for a WAV audio of 100,000, the size reduction results in 10,000 (or when compared to a compact disc the size

reduction is approximately 100,000). While much larger than the seed approach, the CDS approach is still advantageous over the audio stream methods of music storage in the prior art.

While both examples have their advantages, it may also be advantageous to combine aspects of each into a hybrid data structure (HDS). For example, the use of some seed values in the data structure, while also incorporating many of the more complex parameters for the CDS example, preferably can provide an appropriate balance between compatibility and efficiency. Depending on the application and context, the balance between these two goals preferably can be adjusted by using a hybrid data structure that is in between the SDS of Fig. 28 and the CDS of Fig. 30.

In the example of Fig. 30, 'Application Number', 'Style/SubStyle', 'Sound Bank/Synth Type', 'Sample Frequency', 'Sample List', 'Key', 'Tempo', 'Instrument', 'State', and 'Parameter' are preferable parameters that are described above in reference to Fig. 28.

'Song Structure' preferably is data that preferably lists the number of instrument types in the song, as well as the number and sequence of the parts in the song.

'Structure' preferably is data that is indexed by part that preferably can include the number and sequence of the sub-parts within that part.

'Filtered Track' preferably is a parameter that preferably can be used to hold data describing the characteristics of an effect. For example, it preferably can indicate a modulation type of effect with a square wave and a particular initial value. As the effect preferably is typically connected with a particular part, this parameter may preferably be indexed by part.

'Progression' preferably is characteristic information for each sub-part. This might include a time signature, number and sequence of SEQs, list of instrument types that may be masked, etc.

'Chord' preferably contains data corresponding to musical changes during a sub-part. Chord vector (e.g., +2, -1, etc.), key note (e.g., F), and progression mode (e.g., dorian ascending) data preferably are stored along with a time stamp.

'Pattern' and the sub-parameters 'Combination', 'FX Pattern', and 'Blocks', all preferably contain the actual block data and effects information for each of the instruments that are used in the song. This data is preferably indexed by the type of instrument.

'Nota Bene' preferably is for specifying instruments or magic notes that will be played differently each time the song is played. This parameter preferably allows the creation of songs that have elements of improvisation in them.

Additional parameters can preferably be included, for example to enable soundbank data associated with a particular song to be embedded. Following this example, when such a CDS is accessed, the sound bank data preferably is loaded into non-volatile memory accessible to a DSP such that the sound bank data may be used during the generation of music output.

Fig. 31 depicts a preferable example flow chart for the CDS approach discussed above. It is similar to Fig. 29, except that at the points in the flow where the Seeds are loaded, determined, updated, and/or stored, there are corresponding references to loading, determining, updating, and/or storing CDS parameter data corresponding to Song Structure, Structure, Filtered Track, Progression, Chord, Pattern, Instrument, State, Parameter, and Nota Bene.

In certain preferred embodiments the Player 10 is accompanied by a companion PC software system designed to execute on a PC system and communicate with Player 10 via a data link (e.g., USB 54, Serial I/O 57, and/or a wireless link such as 802.11b, Bluetooth, IRDA, etc.). Such a PC software system preferably is configured to provide the user with a simple and effective way to copy files between the Player 10 and other locations (e.g., the PC hard drive, the Internet, other devices, etc.). For example, the companion PC software program preferably operates under the MS Windows family of Operating Systems and provides full access to the User for all Player10 functions and Modes, as well as the local Player memory (e.g., SMC). Following this example, a user can connect to the Internet and upload or download music related files suitable to be used with the Player 10 (e.g., MIDI, WMA, MP3, Karaoke, CDS, SDS, etc.) as well as user interface-related files such as customized user-selectable graphics preferably to be associated with music styles or songs on the Player 10. Such a companion PC program preferably is also used to enable hardware and/or software housekeeping features to be easily managed, such as firmware and sound bank updates. This companion PC software system preferably is used to provide the user with an easy way to share music components and/or complete songs with other users in the world (e.g., via FTP access, as attachments to email, via peer-to-peer networking software such as Napster, etc.). It is important to note the potentially

royalty-free nature and extreme size efficiency of musical output from the Player 10 lends itself well to the Internet context of open source file sharing.

Various examples of preferred embodiments for hardware implementation examples of the present invention will now be described.

Fig. 32 is a block diagram of one portable hardware device embodiment 35 of the present invention. The microprocessor (MP 36) controls local address and data busses (MP Add 37 and MP Data 38); the universal serial bus interface (USB 39), the smart media card interface (SMC 40) (as discussed previously, alternatives to SmartMedia, such as other types of Flash or other memory cards or other storage media such as hard disk drives or the like may be used in accordance with the present invention), and a memory such as Flash 41 are preferably on the MP data bus 38; and the MIDI/Audio DSP (DSP 42) is preferably on both the MP address bus 37 and MP data bus 38. The SMC interface 40 preferably has a buffer 59 between it and the MP Data bus 38, and there preferably are keyboard interface 42 (with MP Data Latch 44) and LCD interface 45 associated with the MP busses as well. In this example, the MP 36 can preferably perform as a sequencer to extract timing information from an input data stream and send MIDI information (possibly including NRPN-type data discussed elsewhere in this disclosure) to the DSP 42. The DSP 42 additionally preferably has dedicated address and data busses (DSP Add 46 and DSP Data 47) that preferably provide access to local RAM 48 and Flash 49 memories.

The MP 36, DSP 42, FM receiver 50, and Microphone input 51 all preferably have some type of input to the hardware CODEC 52 associated with the DSP 42.

The connector 53 at the top left of Fig. 32 can be considered as a docking station interface or as a pure USB interface or external power interface, preferably complete with interfaces for USB 54, power 55, rechargeable battery charge 56, serial I/O 57, and Audio I/O 58. An example of a block diagram for a docking station device 70 of the present invention is provided in Fig. 34. As is shown in Fig. 34, the docking station 70 preferably includes a local microprocessor (LMP 71), preferably with a USB interface 72, address and data busses (LMP ADD 73 and LMP Data 74), a MIDI I/O interface 75, and memory such as Flash 76. Additionally, the docking station device 70 preferably contains an Audio Codec 77, a Video I/O interface 78, and a Power Supply 79.

The MP 36 in this example is preferably the ARM AT91R40807, though any similar microprocessor could be utilized (such as versions that have on-board Flash, more RAM, faster clock, lower voltage/lower power consumption, etc.). This ARM core has 2 sets of instructions: 32bit and 16bit. Having multiple width instructions is desirable in the given type of application in that the 16bit work well with embedded systems (Flash, USB, SMC, etc.), and 32bit instructions work efficiently in situations where large streams of data are being passed around, etc. Other variations of instruction bit length could easily be applied under the present invention.

For 32bit instructions, the system of the present invention preferably pre-loads certain instructions from the Flash memory 41 into the internal RAM of the MP 36. This is because the Flash interface is 16bit, so to execute a 32bit instruction takes at least 2 cycles. Also, the Flash memory 41 typically has a delay associated with read operations. In one example, the delay is approximately 90ns. This delay translates into the requirement for a number of inserted wait states (e.g., 2) in a typical read operation. Conversely, the internal RAM of the MP 36 has much less delay associated with a read operation, and so there are less wait states (e.g., 0). Of course, the internal RAM in this case is 32bits wide, and so the efficiencies of a 32bit instruction can be realized.

As is shown above in the example regarding the wait states of Flash memory 41, there are many reasons why it is desirable to try to maximize the use of the internal MP RAM. As can be seen from Fig. 32, this example of the present invention preferably does not include an SDRAM or RDRAM. While these types of memory means are available to include in such a system, and such use would not depart from the spirit and scope of the present invention, in certain portable applications, such as depicted in Fig. 32, the use of relatively unnecessary complexity (e.g., SDRAM controllers & address logic, etc.) is not preferable. The current example of Fig. 32 achieves many of the benefits of the present invention, in a simple design suitable for a portable device.

One example of a trade-off associated with complexity and portability is the use of a widely available WMA audio decoder algorithm from Microsoft. In this example, when operating the ARM MP of Fig. 32 at 32MHz/3.0V, Microsoft's WMA decoding algorithms can be incorporated to successfully decode and play a WMA-encoded song in stereo at 44KHz and at a sample rate of 128Kbps. However, as discussed elsewhere in this specification, a preferable

feature that allows the speed of an audio stream song to be adjusted can also be incorporated. In this case, when speeding up the WMA 44KHz song using the speed control, it is possible that the system of Fig. 32 may encounter an underrun condition. In this specific example, such cases do not occur when the ARM MP 36 is operated at 40MHz/3.0V. However, when operating the MP 36 at 3.0V, a significant performance hit on battery life can occur. So, because the use of the WMA at 44KHz in combination with the pitch speed feature seems to be relatively unnecessary, this particular example feature can preferably be sacrificed for the benefit of a longer battery life. Obviously, one could incorporate variations such as: a better battery system, a speed stepped approach that operates at full speed when plugged in and at a slower speed when using batteries, a more efficient WMA algorithm, etc. However, this example illustrates the point that competing needs can preferably be balanced with performance and portability.

In the example of Fig. 32, the MP 36 contains 136KB of internal RAM. The performance/portability balance described above dictates that one preferably must play certain tricks on the system to maximize the efficiency of the 136Kb RAM. For example, the memory range can preferably be divided into different regions for buffering, programs, etc., and in real-time modes (e.g., WMA playback), the percentage used for the code can preferably be maximized and the percentage used for buffers preferably minimized.

Another alternative embodiment can be an MP 36 with preferably more internal RAM (for example, 512KB) which would preferably allow a reduction or elimination of the use of Flash memory 41. Such a system may add to the total cost, but would reduce the complexities associated with using Flash memory 41 discussed above.

Another variation is the example shown in Fig. 33, which describes the local DSP area of Fig. 32 wherein preferably additional RAM 90 is accessible on the DSP bus. Such additional RAM can be preferably used to temporarily store large MIDI sound loops that can be played quickly and often. RAM 90 can also preferably be used to temporarily store one or more sound streams (e.g., PCM) that can thus be preloaded and played quickly. Without this feature, each sample might need to be managed and sent by the MP to the DSP every time it is used, in real time. While this is not a problem in certain implementations of the present invention, it may be advantageous to use such additional RAM 90 as shown in Fig. 33 when extensive usage of sound streams is desired. In such cases, a typical size of the RAM 90 in Fig. 33 might preferably be

512KB, and the MP will preferably only need to send an instruction to the DSP to play the locally stored stream.

Continuing the discussion of the architecture shown in Fig. 32, Fig. 35 describes one example for an address map for the internal RAM of the MP. Starting from the bottom of the map, the bottom two sections represent the libraries and routines that are often used, and are always loaded in RAM. The midsection labeled "multi-use" is preferably used for WMA/MP3 related code during the playback of WMA, MP3, and/or other similarly encoded audio stream songs from the SMC. However, during other modes, such as eDJ mode, this midsection is preferably used for Block, Song, and SMC buffers. The next section above this area is preferably used as a buffer for streaming media. This section is preferably divided into a number of subsections, and each subsection is preferably sent to the DSP device at regular intervals (e.g., 5.8ms @44.1kHz, 16bit, 1Kb blocks). Above this, at the top of Fig. 35, is the general-purpose area of MP RAM preferably used for variables and general buffers.

In this example, when the Player is not operating in a WMA/MP3/etc. mode, the 'multi-use' mid section can preferably be used for at least three types of buffers. Block buffers are preferably used by the eDJ Block creation algorithms (e.g., Figs. 24 and 25) to store Block data during operation. Song buffers are preferably used by the eDJ algorithms to store Song data (see Fig. 15) after Block creation has occurred. This Song data is preferably fed out to the DSP device shown in Fig. 32. SMC buffers are preferably used for write operations to the SMC.

SMC is a Flash memory technology that doesn't allow the modification of a single bit. To perform a write to the SMC, one must read the entire SMC Block, update the desired portion of the SMC Block, and then write the entire SMC Block back to the SMC. In the interests of efficiency, the currently used SMC Block is preferably maintained in the SMC buffers.

As one can appreciate, the system configuration described above cannot simultaneously playback large WMA/MP3 streams while also writing to the SMC. This is because the two functions preferably alternatively use the same memory region. This is a creative use of limited resources, because it is preferably a relatively unusual condition to be reading WMA/MP3 while writing SMC at the same time. So the code is preferably arranged to swap in and out of the same location. Such an arrangement allows maximized use of the limited resources in a portable environment such as Fig. 32.

However, in a more powerful environment (with additional resources, and/or faster clock speed), this 'multi-use' of a shared region of memory could preferably be eliminated, and simultaneous use of WMA/MP3 and the Record function could easily be implemented. Obviously, these additional enhancements for use in a portable environment do not limit the other aspects of the present invention.

The system discussed above is portable, but preferably has extremely high-quality sound. On a very basic level, this is partly due to the use of a sound chip that typically would be found in a high-end sound card in a PC system. The SAM9707 chip is preferable because of its excellent sound capabilities, but this has required it be adapted somewhat to work in the portable example discussed herein.

One characteristic of the SAM9707 is that it is typically configured to work with SDRAM in a sound card. This SDRAM would typically hold the MIDI sound banks during normal operation. Such sound banks are preferably a critical part of the final sound quality of music that is output from a DSP-enabled system. In fact, another reason why this particular chip is preferable is to allow custom sounds to preferably be designed.

In the example above of a portable system, SDRAM adds significantly to the power requirements, as well as the address logic. Accordingly, it is desirable to use a variation of the configuration, preferably using Flash as local DSP sound bank storage (see Fig. 32). The use of Flash memory as local DSP storage is a bit problematic because, in order to allow a user to upgrade the sound banks of their portable Player system, the local DSP Flash memory preferably needs to be accessible from the MP side of the architecture. Such access could be gained through the use of a dual-port Flash memory, with memory access from both the DSP busses and the ARM MP busses, but such a dual port architecture would add expenses and complexity to the system.

The problem of reaching a proper balance between maintaining the low power/simple architecture on one hand, and providing high quality, upgradeable, music sound banks on the other hand, is preferably solved by adapting a mode of the DSP chip, and preferably customizing the address logic in such a way that the DSP can be "tricked" into providing the access from the MP side to the local DSP Flash memory.

Fig. 36 describes an example of an addressing space for the DSP local RAM and Flash storage. Starting from the bottom of the map, the first section is preferably for Firmware, and this is typically addressed to a Flash memory region. The next section is preferably the sound banks, and this is also typically addressed to a Flash region. The third section is preferably addressed to Flash when signal A24 is active (in this case, A24 is active low, or = 0). Signal A24 is discussed more below. The fourth section, with starting address 0x1000000, is preferably a 32Kb block that is not addressed to any memory locations. The fifth section is preferably also 32Kb and is preferably addressed to the local DSP RAM (labeled RAM_a). Note that when addressing this area, signal A24 is preferably high. The seventh section, with starting address 0x2000000, is preferably a 32Kb section that preferably resolves to RAM (labeled RAM_b). The two 32Kb RAM regions are preferably combined into the 64Kb local RAM.

So the first variation of the present invention, to the general use of the DSP chip, especially in its intended context of a sound card for a PC, is the address location of the RAM_a. This region is selected to allow a very simple address decode logic arrangement (preferably external to the DSP) so that the assertion of A24 will preferably toggle the destination of RAM_a addresses, between DSP-local RAM and DSP-local Flash memories. This variation preferably involves a firmware modification that will allow the specific location of RAM_a to be configured properly preferably by default at startup time. There are other ways to modify this location after initialization, but they are more complicated, and therefore are not as desirable as the present method.

Another variation to the intended context of the DSP chip address map preferably involves a creative implementation of the DSPs BOOT mode to allow the sound banks to be upgraded, even though the sound banks are preferably located in the local Flash memory of the DSP chip; a location not typically accessible for sound bank upgrades.

In this example, the BOOT mode of the DSP causes an internal bootstrap program to execute from internal ROM. This bootstrap program might typically be used while upgrading the DSP firmware. As such, the internal bootstrap expects to receive 256 words from the 16bit burst transfer port, which it expects to store at address range 0100H-01FFH in the local memory, after which the bootstrap program resumes control at address 0100H. This relatively small burst is fixed, and is not large enough to contain sound banks. Furthermore, it does not allow the

complex Flash memory write activities, as discussed above in connection with the SMC. Since our design preferably uses Flash instead of SDRAM, we have found it highly desirable to use this bootstrap burst to load code that preferably 'tricks' the ROM bootstrap to effectuate the transfer of special code from the ARM MP bus to the RAM. This special code is then used to preferably effectuate the transfer of sound bank upgrade data from the ARM MP bus to the Flash memory.

Fig. 37 is a simple truth table that provides additional information on this unusual use of the DSP bootstrap mode addressing scheme. Fig. 38 is a more detailed truth table that highlights the usefulness of our unusual DSP address logic, including the preferable use of the A24 signal controllable by the ARM MP, preferably by use of the BOOT signal.

In the present example, the A24 address line generated by the DSP is preferably altered by the BOOT signal controlled by the MP before being presented to the address decoding logic of the DSP local memory. This arrangement permits the MP to preferably invert the DSP's selection of RAM and Flash in BOOT mode, and thus allows the RAM to preferably be available at address 0x100 to receive the upgrade code.

Additional variations to the hardware arrangement discussed above can be considered. For example, if the power level is increased, and the MP performance increased, the DSP could be substituted with a software DSP. This may result in lower quality sounds, but it could have other benefits that outweigh that, such as lower cost, additional flexibility, etc. The DSP could similarly be replaced with a general-purpose hardware DSP, with the result of lower quality sounds, possibly outweighed by the benefits of increased portability, etc. The MP could be replaced with one having a greater number of integrated interfaces (e.g., USB, SMC, LCD, etc.), and/or more RAM, faster clock speed, etc. With a few changes to some of the disclosed embodiments, one could practice the present invention with only a DSP (no separate MP), or a dual die DSP/MP, or with only an MP and software. Additionally, the SMC memory storage could be substituted with a Secure Digital (SD) memory card with embedded encryption, and/or a hard disk drive, compact flash, writeable CDROM, etc., to store sound output. Also, the LCD could be upgraded to a color, or multi-level gray LCD, and/or a touch-sensitive display that would preferably allow another level of user interface features.

Yet a further variation of the present discussion preferably can be the incorporation of a electromagnetic or capacitive touch pad pointing device, such as a TouchPad available from

Synaptics, to provide additional desirable characteristics to the user interface. Both the touch pad and the touch sensitive display mentioned above can be used to provide the user with a way to tap in a rhythm, and/or strum a note/chord. Such a device preferably can be used to enable a closer approximation to the operation of a particular instrument group. For example, the touch pad can be used to detect the speed and rhythm of a user's desired guitar part from the way the user moves a finger or hand across the surface of the touch pad. Similarly, the movement of the users hand through the x and y coordinates of such a pointing device can be detected in connection with the pitch and/or frequency of an instrument, or the characteristics of an effect or sample. In another example, a touch pad pointing device can also be used to trigger and/or control turntable scratching sounds approximating the scratching sounds a conventional DJ can generate with a turntable.

As can be seen in Fig. 32, one example of a DSP that can be used in the context of the present invention is the SAM9707 chip available from the Dream S.A. subsidiary of Atmel Corporation. This particular chip is able to handle incoming MIDI and audio stream information.

When incorporating the DSP into a generative/interactive music system, it is highly desirable to synchronize the MIDI and audio streams. A sample preferably has to play at exactly the right time, every time; when the audio stream components get even slightly out of sync with the MIDI events, the resulting musical output generally is unacceptable. This delicate nature of mixing audio streams and MIDI together in a generative/interactive context is worsened by the nature of the Flash read process, in that SMC technology is slow to respond, and requires complex read machinations. It is difficult to accurately sync MIDI events with playback of audio from a Flash memory location. Because of the delay in decoding and playing a sample (compared to a MIDI event), there is a tradeoff in either performing timing compensation, or preloading relatively large data chunks. Because of these issues, it is preferable to configure a new way to use MIDI and audio streams with the DSP chip. While this aspect of the present invention is discussed in terms of the DSP architecture, it will be obvious to one of ordinary skill in the art of MIDI/audio stream synchronization that the following examples apply to other similar architectures.

Fig. 39 shows a simplified logical arrangement of the MIDI and Audio Streams in the music generation process. The two inputs going to the Synth are preferably merged and turned

into a digital audio output signal. This output signal is then preferably fed to a digital to analog converter (DAC), from which is preferably output an analog audio signal suitable for use with headphones, etc. Note that in our example, the Audio stream input to the Synth might typically come from a relatively slow memory means (e.g.; Flash memory), while the MIDI input to the Synth might come from a relatively fast memory means (e.g.; SRAM buffer).

The two inputs to the Synth device preferably may actually share a multiplexed bus; but logically they can be considered as separately distinguishable inputs. In one example, the two inputs share a 16bit wide bus. In this case, the MIDI input preferably may occupy 8bits at one time, and the audio stream input preferably may occupy 16bits at another time. Following this example, one stream preferably may pause while the other takes the bus. Such alternating use of the same bus can mean that relatively small pauses in each stream are constantly occurring. Such pauses are intended to be imperceptible, and so, for our purposes here, the two streams can be thought of as separate.

Fig. 40 shows a simplified MIDI/Audio Stream timeline. Assume that Fig. 40 is the timing for the very beginning of a Block. It follows then, that in this case, the designer wants to play a MIDI note, starting 250ms after the beginning of the Block, that will last 500ms. The duration of the note relates to the type of note being played, for example, if it is a quarter note in a 4/4 time, and with a measure duration of 2 seconds, a 500ms would correspond to a quarter note duration. Also indicated in Fig. 40, that an Audio stream event such as a short voice sample "yo" will preferably be synchronized to occur in the middle of the MIDI event. Bear in mind that this method allows the sample to preferably be quantized to the music, in the sense that it can involve the subtle correction of minor timing errors on the part of the user by synchronizing the sample to the musical context.

In this example, largely because of the constraints of the system architecture example discussed above, this is not a trivial thing to accomplish consistently and accurately using conventional techniques. Keeping in mind that the MIDI event is preferably generated almost instantly by the Synth chip, whereas the Audio Stream event could require one or more of the following assistance from the ARM MP: fetching a sound from SMC, decompressing (PCM, etc.), adding sound effects (reverb, filters, etc.).

In this example, it is highly desirable to create a special MIDI file preferably containing delta time information for each event, and specialized non-registered parameter numbers (NRPNs). This feature is especially advantageous when used with a Sample List (as mentioned above) because the name of a particular sample in a list is preferably implicit, and the NRPNs can preferably be used to trigger different samples in the particular sample list without explicitly calling for a particular sample name or type. This type of optimization reduces the burden of fetching a particular sample by name or type, and can preferably allow the samples used to be preloaded.

Fig. 41 depicts an example of a MIDI NRPN that can be advantageously incorporated into the present invention to allow efficient synchronization of MIDI events with audio samples and effects. The left column depicts the hexadecimal values making up the MIDI NRPN stream. As anyone who works with the MIDI Specification (previously incorporated by reference) will appreciate, the MIDI NRPN is a data structure that enables custom use of portions of a MIDI stream. Accordingly, it can preferably be used to trigger specific custom events for a given architecture.

In Fig. 41, the first hexadecimal value 'B0' preferably indicates a channel number, as well as that it is a MIDI controller command. This can be used to assist with routing in a multi-channel arrangement. In our example, for purposes of simplicity this is set channel 0. The second value '63' preferably indicates that this particular stream contains NRPN information for a particular controller (e.g., 'A'). In this example, NRPN Controller A can be understood by the firmware/software to indicate an audio sample type. The third row value of '40' preferably is data that corresponds to the controller, and in our example this data can be understood to describe the type of sample. As an example of the usefulness of this arrangement, if the type is set to 'long', then the firmware/software preferably can arrange to load the sample in chunks. The fourth row preferably indicates a delta time, in MIDI clicks, that can preferably be used to precisely time the next event. In our example, this delta time is set to '00' for simplicity. The fifth row preferably indicates that this particular stream contains NRPN information for a 'B' controller. In this example, NRPN Controller B can be understood by firmware/software to indicate an audio effects type. This is because we have found it advantageous to use a MIDI DSP component that includes certain audio effects that can be controlled effectively in a timely

manner via MIDI NRPNs. The sixth row preferably indicates the identification of the particular audio effects type called for in this NRPN example. While '00' is shown for simplicity, it should be understood that the value in this part of the MIDI stream can be interpreted by the firmware/software to select a particular effect from the available audio effects for a particular architecture. The seventh row preferably indicates another delta time that can be interpreted as a delay. The eighth row preferably can be used to indicate to the firmware/software the identification of a register to store the NRPN Controller A value shown in row nine. The ninth row uses '03' as an example; this preferably can be interpreted to mean the third audio sample in a list corresponding to a song (see 'Sample List' in Figs. 29 and 30). Value '00' can be used effectively to instruct the firmware/software to select a sample from the sample list randomly. The tenth row of Fig. 41 is preferably another delta time value (e.g., '00' is zero MIDI clicks). The eleventh row preferably can be used to indicate to the firmware/software the identification of a register to store the NRPN Controller B value shown in row 12. The twelfth row uses '07' as an example; in the present discussion this preferably can be interpreted by the firmware/software to instruct the MIDI DSP to apply a particular audio effect among those available.

Fig. 42 is a simplified depiction of a special MIDI type file that is an example of the arrangement of the data being sent from the ARM MP to the DSP preferably via the MIDI input stream, along the lines of the example above.

The top of the figure indicates that the first information in this file is a delta time of 250ms. This corresponds to the 250ms delay at the beginning of Fig. 40. Next in the file depicted in Fig. 42 is general MIDI information preferably indicating a note on event for channel 1, pitch C. This corresponds to the time in Fig. 40 when 250ms has passed. Next in Fig. 42, we have another 250ms delta time. This represents the time between the previous MIDI event, and the next Audio Stream event at time 500ms in Fig. 40. Next, in Fig. 42 we have an NRPN message that preferably indicates to the Synth chip that it needs to play the audio stream event X, with various parameters P, and various effects E. This corresponds to the audio stream event ('yo') depicted in Fig. 40. Then, in Fig. 42 we have another delta time event of 250ms, followed by the general MIDI information preferably indicating a note off event for channel 1, pitch C. This final step corresponds to the end of the MIDI event in Fig. 40 (e.g., 'C' quarter note).

In the previous example, the delta time preferably can be different (and often is) each time in the special MIDI type file. In our simplified example, and because we want to make the timing relationship with a quarter note, etc., more clear, we have used the same 250ms value each time. Obviously, in a more complex file, the delta time will vary.

As previously described, voice and other audio samples may be encoded, stored and processed for playback in accordance with the present invention. In certain preferred embodiments, voice samples are coded in a PCM format, and preferably in the form of an adaptive (predictive), differential PCM (ADPCM) format. While other PCM formats or other sample coding formats may be used in accordance with the present invention, and particular PCM coding formats (and ways of providing effects as will be hereinafter described) are not essential to practice various aspects of the present invention, a description of exemplary ADPCM as well as certain effects functions will be provided for a fuller understanding of certain preferred embodiments of the present invention. In accordance with such embodiments, a type of ADPCM may provide certain advantages in accordance with the present invention.

As will be appreciated by those of skill in the art based on the disclosure herein, the use of ADPCM can enable advantages such as reduced size of the data files to store samples, which are preferably stored in the non-volatile storage (e.g., SMC), thus enabling more samples, song lists and songs to be stored in a given amount of non-volatile storage. Preferably, the coding is done by a packet of the size of the ADPCM frame (e.g., 8 samples). For each packet, preferably a code provides the maximum value; the maximum difference between two samples is coded and integrated in the file. Each code (difference between samples (delta_max) and code of the packet (diff_max)) uses 4 bits. In accordance with this example, the data/sample is therefore (8*4+4)/8 = 4.5 bits/sample.

As will be appreciated, this type of coding attempts to code only what is really necessary. Over 8 samples, the maximum difference between two samples is in general much less than the possible dynamic range of the signal (+32767/-32768), and it is therefore possible to allow oneself to code only the difference between samples. Preferably, the ADPCM is chosen to be suitable for the voice that is relatively stationary. By predictive filtering, it is possible to reduce the difference between a new sample and its prediction. The better the prediction, the smaller the difference, and the smaller the coding (the quantization) that is chosen, taking into account the

average differences encountered. While it will be appreciated that this approach requires additional computation ability for the prediction computation, it is believed that this approach provides significant advantages in reduced storage for samples with acceptable sample coding quality in accordance with the present invention. While more conventional or standardized ADPCM desires to offer a coding time without introducing delays, with the present invention it has been determined that such attributes are not essential.

A simple coding without prediction and taking into account only average values of differences encountered reacts very poorly to a non-stationary state (e.g., each beginning of a word or syllable). For each new word or syllable, a new difference much greater than the average differences previously encountered typically cannot be suitably coded. One therefore tends to hear an impulse noise depending on the level of the signal. Preferably, the solution is therefore to give the maximum value of the difference encountered (one therefore has a delay of 8 samples, a prediction is thus made for the quantizer only) for a fixed number of samples and to code the samples as a function of this maximum difference (in percentage). The coding tends to be more optimal at each instant, and reacts very well to a non-stationary state (each beginning of a word or syllable). Preferably, the coding is logarithmic (the ear is sensitive to the logarithm and not to the linear), and the Signal/Noise ratio is 24 db. In preferred embodiments, this function is put in internal RAM in order to be executed, for example, 3 times more rapidly (one clock cycle for each instruction instead of three in external flash memory).

Preferably certain effects may be included in the ADPCM coding used in certain embodiments of the present invention. For example, a doppler effect may be included in the ADPCM decoding since it requires a variable number of ADPCM samples for a final fixed number of 256 samples. As is known, such a doppler effect typically consists of playing the samples more or less rapidly, which corresponds to a variation of the pitch of the decoded voice accompanied by a variation of the speed together with the variation of pitch. In order to give a natural and linear variation, it is desirable to be able to interpolate new samples between two other samples. The linear interpolation method has been determined to have certain disadvantages in that it tends to add unpleasant high frequency harmonics to the ear.

The method traditionally used consists of over-sampling the signal (for example, in a ratio [of] 3 or 4) the signal and then filtering the aliasing frequencies. The filtered signal is then

interpolated linearly. The disadvantage of this method is that it requires additional computational ability. Preferably, in accordance with certain embodiments, a technique is utilized that consists of interpolating the signal with the four adjacent samples. It preferably corresponds to a second order interpolation that allows a 4.5 dB gain for the harmonics created by a linear interpolation. While 4.5 db seems low, it is important to consider it in high frequencies where the voice signal is weak. The original high frequencies of the voice are masked by the upper harmonics of the low frequencies in the case of the linear method, and this effect disappears with second order interpolation. Moreover, it tends to be three times faster than the over-sampling method. Preferably, this function is put in internal RAM in order to be executed, for example, 3 times more rapidly (one clock cycle for each instruction instead of three in external flash memory).

Also in accordance with preferred embodiments, an electronic metronome function is included, which consists of counting the period number (the pitch) in an analysis window in order to deduce from this the fundamental frequency. Preferably, this function may be utilized to process samples in order to reveal the periods. In general, it is not feasible to count the peaks in the window because the signal tends to vary with time (for example, the beating of 1 to 3 piano strings that are not necessarily perfectly in tune); moreover, in the same period, there can be more than one peak. In accordance with such embodiments, the distance between a reference considered at the beginning of the analysis window and each of the panes shifted by one sample. For a window of **2*WINDOW_SIZE** samples and a reference window of **WINDOW_SIZE** samples, one therefore may therefore carry out **WINDOW_SIZE** computations of distance on **WINDOW_SIZE** samples. Preferably, the computation of distance is done by a sum of the absolute value of the differences between reference samples and analysis samples. This function preferably is put in internal RAM in order to be executed, for example, 3 times more rapidly (one clock cycle for each instruction instead of three in external flash memory).

Also in accordance with such embodiments, special effects such as wobbler, flange, echo and reverb may be provided with the ADPCM encoding. Such special effects preferably are produced over 256 samples coming from the ADPCM decoder and from the doppler effect. Preferably, this function is put in internal RAM in order to be executed, for example, 3 times more rapidly (one clock cycle for each instruction instead of three in external flash memory). Preferably, the average value of the sample is computed, and it is subtracted from the sample

(which can be present over the samples) in order to avoid executing the wobbler function on it, which would add the modulation frequency in the signal (and tend to produce an unpleasant hiss). Preferably, the method for the wobbler effect is a frequency modulation based on sample = sample multiplied by a sine function (based on suitable wobbler frequencies, as will be understood by those of skill in the art).

Also in accordance with the preferred embodiments, the purpose of the flange effect is to simulate the impression that more than one person is speaking or singing with a single source voice. In order to limit the computation power, two voices preferably are simulated. In order to provide this impression, preferably the pitch of the source voice is changed and added to the original source voice. The most accurate method would be to analyze the voice using a vocoder and then to change the pitch without changing the speed. In each case, one could have the impression that a man and a woman are singing together, although such a method typically would require DSP resources. A method that changes the pitch without changing the speed (important if one wants the voices to remain synchronous) consists of simulating the second voice by alternately accelerating and decelerating the samples. One then produces the doppler effect explained in the preceding, but with a doppler that varies alternately around zero in such a way as to have a slightly different pitch and the voices synchronous. With such embodiments, one may simulate, for example, a person placed on a circle approximately 4 meters in diameter regularly turning around its axis and placed beside another stationary person.

Also in accordance with such embodiments, the echo effect is the sum of a source sample and of a delayed sample, and the reverb effect is the sum of a source sample and a delayed sample affected by a gain factor. The delayed samples preferably may be put in a circular buffer and are those resulting from the sum. The formula of the reverb effect may therefore be:

Sample(0) = sample(0) + sample(-n)*gain + sample(-2*n)*gain^2 + sample (-3*n)*gain^ + ... + sample(-i*n)*gain^i. Preferably, the gain is chosen to be less than 1 in order to avoid a divergence. In accordance with preferred embodiments, for reasons of size of the buffer, which can be considerable, the echo effect preferably uses the same buffer as that of the reverb effect. In order to have a true echo, it is necessary to give reverb a gain effect that is zero or low. The two effects can function at the same time. The delay between a new sample and an old one is produced by reading the oldest sample put in the memory buffer. In order to avoid shifting the

buffer for each new sample, the reading pointer of the buffer is incremented by limiting this pointer between the boundaries of the buffer. The size of the memory buffer therefore depends on the time between samples.

Also in accordance with such embodiments, an electronic tuner function may be provided, the aim of which is to find the fundamental of the sample signal coming from the microphone in order to give the note played by a musical instrument. Similar to what has been described previously, a preferred method will consist of computing the number of periods for a given time that is a multiple of the period in order to increase the accuracy of computation of the period. In effect, a single period will give little accuracy if the value of this period is poor because of the sampling. In order to detect the periods, preferably one uses a routine which computes the distance between a reference taken at the beginning of the signal and the signal. As will be understood, the period will be the position of the last period divided by the total number of periods between the first and the last period. The effective position of the last period is computed by an interpolation of the true maximum between two distance samples. The period thus computed will give by inversion (using a division of 64 bits/32bits) the fundamental frequency with great precision (better than 1/4000 for a signal without noise, which is often the case).

Also in accordance with such embodiments, a low pass filter (or other filter) function may be provided as part of the effects provided with the ADPCM sample coding. Such a function may eliminate with a low-pass filter the high frequencies of the samples used for computation of the distance such for the routines previously described. These high frequencies tend to disturb the computations if they are too elevated. Filtering is done by looking for the highest value in order to normalize the buffer used for computation of the distance.

Also in accordance with the present invention, there are numerous additional implementations and variations that preferably can be used with many desirable aspects of the present invention. Exemplary ways to use the present invention to great effect include a software-based approach, as well as general integration with other products. Additionally, several valuable variations to the present invention can be used with great success, especially with regard to media content management, integration with video, and other miscellaneous variations.

Many aspects of the present invention can be incorporated with success into a software-based approach. For example, the hardware DSP of the above discussion can be substituted with a software synthesizer to perform signal processing functions (the use of a hardware-based synthesizer is not a requirement of the present invention). Such an approach preferably will take advantage of the excess processing power of, for example, a contemporary personal computer, and preferably will provide the quality of the music produced in a hardware-based device, while also providing greater compatibility across multiple platforms (e.g., it is easier to share a song that can be played on any PC). Configuring certain embodiments of the present invention into a software-based approach enables additional variations, such as a self-contained application geared toward a professional music creator, or alternatively geared towards an armchair music enthusiast. Additionally, it is preferable to configure a software-based embodiment of the present invention for use in a website (e.g., a java language applet), with user preferences and/or customizations to be stored in local files on the user's computer (e.g., cookies). Such an approach preferably enables a user to indicate a music accompaniment style preference that will 'stick' and remain on subsequent visits to the site. Variations of a software-based approach preferably involve a 'software plug-in' approach to an existing content generation software application (such as Macromedia Flash, Adobe Acrobat, Macromedia Authorware, Microsoft PowerPoint, and/or Adobe AfterEffects). It is useful to note that such a plug-in can benefit from the potentially royalty free music, and that in certain embodiments, it may be preferable to export an interactively generated musical piece into a streaming media format (e.g., ASF) for inclusion in a Flash presentation, a PDF file, an Authorware presentation, an AfterEffects movie, etc. Certain embodiments of the present invention can be involved in a Internet-based arrangement that enables a plurality of users to interactively generate music together in a cooperative sense, preferably in real time. Aspects of the present invention involving customized music can be incorporated as part of music games (and/or music learning aids), news sources (e.g., internet news sites), language games (and/or language learning aids), etc. Additionally, a software/hardware hybrid approach incorporating many features and benefits of the present invention can involve a hybrid "DSP" module that plugs into a high speed bus (e.g., IEEE 1394, or USB, etc.) of a personal computing system. In such an approach, the functionality of MP 36 can be performed by a personal computing system, while the functionality of DSP 42 can be

performed by a DSP located on a hardware module attached to a peripheral bus such as USB. Following this example, a small USB module about the size of a automobile key can be plugged into the USB port of a PC system, and can be used to perform the hardware DSP functions associated with the interactive auto-generation of algorithmic music.

As will be appreciated, aspects of the present invention may be incorporated into a variety of systems and applications, an example of which may be a PBX or other telephone type system. An exemplary system is disclosed in, for example, USP 6,289,025 to Pang et al., which is hereby incorporated by reference (other exemplary systems include PBX systems from companies such as Alcatel, Ericsson, Nortel, Avaya and the like). As will be appreciated from such an exemplary system, a plurality of telephones and telephony interfaces may be provided with the system, and users at the facility in which the system is located, or users who access the system externally (such as via a POTS telephone line or other telephone line), may have calls that are received by the system. Such calls may be directed by the system to particular users, or alternatively the calls may be placed on hold (such aspects of such an exemplary system are conventional and will not be described in greater detail herein). Typically, on-hold music is provided to callers placed on hold, with the on-hold music consisting of a radio station or taped or other recorded music coupled through an audio input, typically processed with a coder and provided as an audio stream (such as PCM) and coupled to the telephone of the caller on hold.

In accordance with embodiments of the present invention, however, one or more modules are provided in the exemplary system to provide on-hold music to the caller on hold. Such a module, for example, could include the required constituent hardware/software components of a Player as described elsewhere herein (see, e.g., Fig. 32 and related description) (for purposes of this discussion such constituent hardware/software components are referred to as an "auto-composition engine"), but with the user interface adapted for the PBX-type of environment. In one such exemplary embodiment, one or more auto-composition engines are provided, which serve to provide the on-hold music to one or more callers on hold. In one example, a single auto-composition engine is provided, and the first caller on hold may initially be presented with auto-composed music of a particular style as determined by the auto-composition engine (or processor controlling the exemplary system) (this may also be a default on hold music style selected by a configuration parameter of the exemplary system). Preferably, via an audio prompt provided by

the resources of the exemplary system, the caller on hold is provided with audio information indicating that the caller on hold may change the style of on-hold music being provided (such audio prompt generation is considered conventional in the context of such exemplary systems and will not be described in greater detail herein). Preferably, the user may indicate such desire by pressing a predetermined digit (which preferably is identified in the audio prompt) on the telephone key pad, which may be detected by the resources of the exemplary system (such digit detection capability is considered conventional in the context of such exemplary systems and will not be described in greater detail herein), and thereafter may be provided with preferably a plurality of music styles from which to select the style of on-hold music (such as with audio prompts providing available styles of music followed by one or more digits to be entered to select the desired style of music). Thereafter, the user may depress the appropriate digit(s) on the telephone keypad, which are detected by the resources of the exemplary system, which preferably decodes the digits and sends control information to one of the auto-composition engines, in response to which the auto-composition engine thereafter begins to auto-compose music of the selected style, which is directed to the caller on hold as on hold music.

What is important is that, in accordance with such embodiments, one or more auto-composition engines are adapted for the exemplary system, with the command/control interface of the auto-composition engine being changes from buttons and the like to commands from the resources of the exemplary system (which are generated in response to calls being placed on hold, digit detection and the like). In accordance with variations of such embodiments, a plurality of auto-composition engines are provided, and the resources of the system selectively provide on-hold music to on hold callers of a style selected by the caller on hold (such as described above). In one variation, there may potentially be more callers on hold than there are auto-composition engines; in such embodiments, the callers on hold are selectively coupled to one of the output audio streams of the auto-composition engines provided that there is at least one auto-composition engine that is not being utilized. If a caller is place on hold at a time when all of the auto-composition engines are being utilized, the caller placed on hold is either coupled to one of the audio streams being output by one of the auto-composition engines (without being given a choice), or alternatively is provided with an audio prompt informing the user of the styles of on-hold music that are currently being offered by the auto-composition engines (in response

thereto, this caller on hold may select one of the styles being offered by depressed one or more digits on the telephone keypad and be coupled to an audio stream that is providing auto-composed music of the selected style).

Other variations of such embodiments include: (1) the resources of the exemplary system detect, such as via caller ID information or incoming trunk group of the incoming call, information regarding the calling party (such as geographic location), and thereafter directs that the on hold music for the particular on hold be a predetermined style corresponding to the caller ID information or trunk group information, etc.; (2) the resources of the exemplary system selectively determines the style of the on-hold music based on the identity of the called party (particular called parties may, for example, set a configuration parameter that directs that their on hold music be of a particular style); (3) the resources of the exemplary system may selectively determine the style of on-hold music by season of the year, time of day or week, etc.; (4) the exemplary system includes an auto-composition engine for each of the styles being offered, thereby ensuring that all callers on-hold can select one of the styles that are offered; (5) default or initial music styles (such as determined by the resources of the exemplary system or called party, etc., as described above) are followed by audio prompts that enable the caller on hold to change the music style; and (6) the resources of the exemplary system further provide audio prompts that enable a user to select particular music styles and also parameters that may be changed for the music being auto-composed in the particular music style (in essence, audio prompt generation and digit detection is provided by the resources of the exemplary system to enable the caller on hold to alter parameters of the music being auto-composed, such as described elsewhere herein.

Other examples of novel ways to generally integrate aspects of the present invention with other products include: video camera (e.g., preferably to enable a user to easily create home movies with a royalty free, configurable soundtrack), conventional stereo equipment, exercise equipment (speed/intensity/style programmable, preferably similar to workout-intensity-programmable capabilities of the workout device, such as a StairMaster series of hills), configurable audio accompaniment to a computer screensaver program, and configurable audio accompaniment to an information kiosk system.

Aspects of the present invention can advantageously be employed in combination with audio watermarking techniques that can embed (and/or detect) an audio 'fingerprint' on the

musical output to facilitate media content rights management, etc. The preferable incorporation of audio watermarking techniques, such as those described by Verance or Digimarc (e.g., the audio watermarking concepts described by Digimarc in US patents 6,289,108 and 6,122,392, incorporated herein by reference), can enable a user with the ability to monitor the subsequent usage of their generated music.

In another example, certain embodiments of the present invention can be incorporated as part of the software of video game (such as a PlayStation 2 video game) to provide music that preferably virtually never repeats, as well as different styles preferably selectable by the user and/or selectable by the video game software depending on action and/or plot development of the game itself.

Additionally, there are certain novel variations to the present invention that incorporate many advantages of the present invention to great effect. For example, in the portable hardware device 35 in Fig. 32, the incoming data on MIC input 51 (e.g., a vocal melody of the user) can pass through hardware codec 52 to MP 36, where it can be analyzed by the MP 36 and processed/adjusted by DSP 42 (under control of MP 36) to subtly 'improve' pitch and/or rhythm characteristics. This example illustrates a preferable arrangement that allows a user's vocal input to be adjusted to conform to the key and/or rhythmic characteristics of the accompanying music. Continuing this example, the pitch of a user's input to MIC input 51 preferably can be analyzed by the portable hardware device 35 and bumped up or down in pitch to more closely match a pitch that fits the current key and/or mode of the music. Such a variation provides a novice user with a easy way to generate songs that are musically compelling, yet preferably are also noticeably derivative of the user's input (e.g., vocal). In another example variation, the circuitry mentioned here preferably can be available to analyze the user's input (e.g., vocal) and infer some type of timing and/or melody information, which information preferably can then be used in the interactive music autogeneration to help define the pitch values and/or the rhythmic data comprised in the RP. This example presents a way for a user to demonstrably interact with, and influence, the musical output, all the while without needing to fully understand the complexities of musical composition.

Additionally, many aspects of the present invention are useful to enable a new concept in Firmware upgrades. Using aspects of the present invention, firmware updates can be made

available to users, complete with embedded advertising, which provides the Firmware manufactures/distributors with a revenue source other than the user. This concept preferably involves the distribution of firmware (or other software-based programs such as sound bank data) upgrades that contain embedded advertising images (and/or sounds). Such images/sounds preferably can temporarily appear during the operation of the music product, and can fund the development of customized firmware for users to preferably freely download.

As will be understood by a person of ordinary skill in the art of portable electronic music design, the examples discussed here are representative of the full spirit and scope of the present invention. Additional variations, some of which are described here, incorporate many aspects of the present invention.

Although the invention has been described in conjunction with specific preferred and other embodiments, it is evident that many substitutions, alternatives and variations will be apparent to those skilled in the art in light of the foregoing description. Accordingly, the invention is intended to embrace all of the alternatives and variations that fall within the spirit and scope of the appended claims. For example, it should be understood that, in accordance with the various alternative embodiments described herein, various systems, and uses and methods based on such systems, may be obtained. The various refinements and alternative and additional features also described may be combined to provide additional advantageous combinations and the like in accordance with the present invention. Also as will be understood by those skilled in the art based on the foregoing description, various aspects of the preferred embodiments may be used in various subcombinations to achieve at least certain of the benefits and attributes described herein, and such subcombinations also are within the scope of the present invention. All such refinements, enhancements and further uses of the present invention are within the scope of the present invention.

To further facilitate the various embodiments of the present invention, and the novel uses and applications thereof (including starting and using a particular implementation of a Player known as the "MadPlayer"), the following documents are attached hereto and incorporated herein by reference as part of this application and invention: MadPlayer MadManual (76 pages); MadPlayer Quickstart and Setup Guide (3 pages); MadWare Reference (26 pages); and Custom Graphics Specification (13 pages).

MADPLAYER

MadManual™

Preliminary Edition:
Information in this Guide is
preliminary and subject to change.

This page intentionally left blank.

TABLE OF CONTENTS

MADPLAYER™ PACKAGE CONTENTS

- The MadPlayer™.
- Headphones + embedded microphone.
- 32MB SmartMedia™ Card.
- 2 "AA" rechargeable batteries, with charger.
- CD containing MadManual™ and MadWare™ PC software.
- MadQuickStart™ Card.
- USB cable (1.5m)

INTRODUCING THE MADPLAYER™

No Way?
The MadPlayer™ is the first and only hand-held digital music player that let's you ***create*** your own music. New. From scratch. Just the way you want it to be.

Get used to it!
Brand new products are brand new ideas. Until you start playing with your MadPlayer you're not going to believe what it can do.

Let it take you there...
Composing. Rapping. Singing. Performing.
Sampling. Playlists. Song sharing. Karaoke.
FM Radio. MP3. WMA. MIDI.
Save hundreds of MadPlayer songs on a single SmartMedia™ card.

What are you waiting for!?
The best way to start having fun with your MadPlayer is to use the MadStart Guide, included in the box, and ***make music!***

Using this Manual

Still here? Then you're looking for the whole story, the big picture, the nuts and bolts. This is it!

Should you read this manual from page one to the end? You can, it takes all kinds. Or you can dip into it anytime you're playing with the MadPlayer and you think there's something more you could be doing in one Mode or another. There will be, bet on it. Never assume the MadPlayer can't do something. Always assume you just have to learn how to get what you want.

If you plan to read this front to back then click the scroll button, or turn the page, and have fun. If you're a dip-when-you-need-to reader look at the table on contents, Page 8, find your topic, and dive in.

For you hard-core audio junkies who want those electronic input/output numbers, exact wavelengths, and the how-to-tweak anything in the MadWorld™ step-by-step instructions...they're all spelled out in the end-of-manual Technical Appendixes, so the rest of us won't have to get nervous every time we see that stuff.

MADPLAYER™ FEATURES

Generactive Music:

Push a button and MadPlayer™ begins composing a song from scratch, in whatever music style you choose. This is called **Generative** music—completely new music created by mathematical algorithms following the rules of a music-style framework—Techno, Hip-Hop, etc. At any point you can **Interact** with the song-in-progress—changing or modifying any aspect of the song through the use of the joystick and control buttons.

Generative Music + Your ***Interaction*** = **Gener*active*** Music.

The "Highway:"

While the MadPlayer composes and plays a song, the LCD screen pictures a six-lane highway that corresponds to the six elements of the song with which you can interact. Those highway "lanes" represent the Drums, Bass, Riff, Lead, Microphone, and Samples. During a song you can "Tunnel" under any "lane" and change or modify that element of the song. Your beat, your sound, your music!

Digital Audio Player:

Download music from the web, your friends, or ripped from your own CDs. Store up to 120 minutes of WMA-format digital music—songs, playlists, samples—when using a 64MB SmartMedia™ Card. Then use your MadPlayer to listen to it anywhere.

FM Radio:

Tune in your favorite radio station on the MadPlayer and crank it up! Hear that? Sample your favorite songs or jingles. Then save them to use in your own generactive MadPlayer songs. Rap to that new beat today!

Microphone/Sampler:

Want to sample? Record and save voice or sound samples anywhere. Then, once they're saved, apply any of five effects—Reverb, Wobbler, Low Voice, Doppler and Smurf. Performing with the microphone? You control it all: change the pitch of your voice, add echo effects, adjust the balance or volume—all in real time, on the fly, without missing a beat.

Karaoke:

Download Karaoke song files to your SmartMedia™ Card. When MadPlayer plays them, it displays the lyrics on its LCD screen, and let's you sing along using full microphone effects. A hand-held party, anywhere!

SmartMedia™ Card Storage:

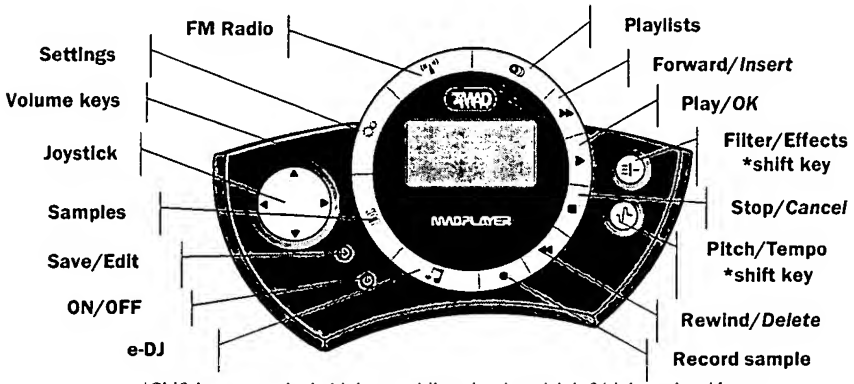
Up to 500 MadPlayer-generated songs can be stored on even the smallest (8MB) SmartMedia™ Card. And the SmartMedia™ Card can also be used to store Samples, WMA, MP3 (converted), Midi or Karaoke Files. [**Important Note**: See

Loading the SmartMedia™ Card on page 12.]

USB Interface:

Fast and easy transfer of songs and samples from/to your PC. Quick access to future firmware upgrades and sound bank downloads.

MADPLAYER™ CONTROLS



*Shift keys must be held down while using joystick left/right and up/down.

Figure 1: MadPlayer™ controls

On/Off (⏻):

Press the button for more than 2 seconds: **MadPlayer is on.**
Press the button briefly to turn the **LCD Backlight on/off.**
Press the button for more than 2 seconds, again: **MadPlayer is off.**

Volume:

Output volume is controlled by a − + key located on the top side of the player. This key is also used with the Effects/Filters or *Pitch/Tempo* shift keys to control microphone volume.

Save/Edit (⇳) :

A *short press* of this key is used to **Save MadSongs**, or to **Enter Edit Mode** when an item has been selected in a menu list of songs, samples, or radio presets. It is also used to **Save Changes** made within certain edit menus.
A *long press* of this key **Locks the Keys** of the MadPlayer. A second *long press* **Unlocks the Keys.**

Player Keys:

The ▶ *play*, ■ *stop*, ▶▶ *forward* and ◀◀ *reverse* keys are used to control the playback of MAD, MIDI or WMA songs, as well as samples from the Sample Menu.

The ● *record* key controls sample recording.

The player keys also have the following functions when used in Selection or Edit Menus:
▶ ***play*** selects a sub menu or validates a change (▶ ***play*** = OK).
■ ***stop*** reverts to previous menu, cancels an action, or discards a change (■***stop*** = Cancel).
▶▶ ***forward*** inserts a list item, or creates a new list (▶▶***forward*** = Insert/Create).
◀◀ ***reverse*** deletes a list item (◀◀ ***reverse*** = Delete).

Mode Keys:

♫ ***e-DJ***, "((♟))" ***Radio***, ◎ ***Songs***, ▤ ***Samples*** and ☼ ***System*** are direct-access keys that switch Modes with one press of the button.

The Joystick, Pitch/Tempo and Effects/Filters(FX) Keys:

The **Joystick** toggles up/down and left/right. It is used for menu navigation, song or music style selection, and for real-time interaction with music.

The √ *Pitch/Tempo* Key controls Music Tempo when pressed simultaneously with Joystick left/right. It controls Music Pitch when pressed simultaneously with Joystick up/down.
When this key is pressed alone, the current pitch and tempo values are displayed in the status line.

The ≡|~ **Effects/Filters (FX) Key** operates when pressed simultaneously with Joystick up/down or left/right. Depending on context, the key can control filter frequency and filter resonance of certain instruments used in MadSongs, as well as sound effects, filters, echo, reverb and volume.
When this key is pressed alone, the active Equalizer preset is displayed in the status line.

Both the Pitch/Tempo and FX keys are referred to as "shift keys," because they modify the role of the joystick when pressed down.

OPERATING THE MADPLAYER™

Warning!

Do not insert the SmartMedia™ Card now! **See instructions on Page 12.**

Insert Batteries:

Open the compartment door and insert two AA alkaline, lithium or rechargeable batteries. [Note: When the battery level becomes too low, a "LOW BATTERY" message appears in the status line (top line of the LCD screen). Batteries should be replaced immediately to avoid a potential loss of power during SmartMedia card write cycle, which could cause data loss and/or damage to the card.]

Turn on MadPlayer™:

Press the ⏻ *On/Off* button for more than 2 seconds: **MadPlayer is On.**
[Press the ⏻ *On/Off* button briefly to turn LCD Backlight on/off.]

Home Mode /Help Mode

Opening Screen:

When MadPlayer is first turned on it is in **Home Mode**. [See Figure 2.] This is the Home Screen graphic for Home Mode. If a SmartMedia™ card has been inserted and has Auto-Play feature ON (see System Configuration on page 46), it will automatically start playing the Demo Songs.

Figure 2: Home Mode Screen

Custom Home screens:

You can replace the standard Home screen with your own, customized static or animated screen! You can download these screens from www.MadPlayer.com or import them from bitmap files (BMP) with a special toolbox included with the MadWare™ PC Applications CD (coming soon!). Some samples are also included on the MadWare™ CD. Just download the custom screen file (*.DBS) to the SmartMedia™Card and see!

Help Screen:

When the joystick is pressed in Home Mode, the main **Help Screen** is displayed, prompting the user to press any key for key-specific Help. [See Figure 3].

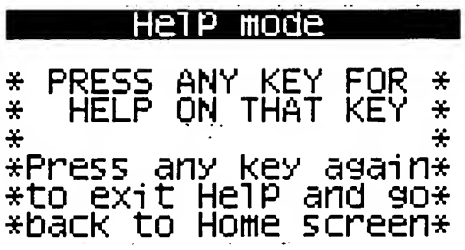


Figure 3: Help Mode Screen

Key-Specific Help Screen:

Press the √ *Pitch/Tempo* key. The Help Text for √ *Pitch/Tempo* is now displayed. [See Figure 4: Key-Specific Help Screen.] To return to Home Mode, press the √ *Pitch/Tempo* key a second time. Use this sequence to get key-specific Help for any key, and to then return to the Home Mode screen.

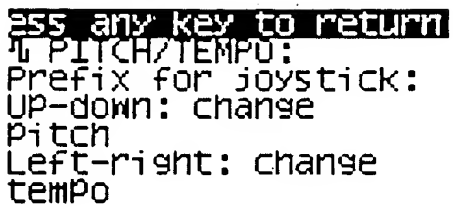


Figure 4: Key-Specific Help Screen

Loading the SmartMedia™ Card

Storage Choices:

Only use 3.3V (also called 3V) SmartMedia™ Cards.
MadPlayer accepts 4MB, 8MB, 16MB, 32MB or 64 MB Cards.
See SmartMedia™ Card technical information, page 74, for more specifics.

Card Loading:

The best time to insert a SmartMedia™ Card is *before* you turn on the MadPlayer, although it can be inserted while the MadPlayer is operating.

Step 1: Turn over the MadPlayer so that the back faces you. Turn the card so that its gold-colored contacts face the back of the MadPlayer, and so that the card end with a notched corner is the end that will be fully inserted. Now slide the card into the slot as far as it will go. [The card will not slide all the way into the slot if it is inserted facing the wrong way. Do not apply force to the card. It should slide in easily.]

Step 2: Once the card is installed and the MadPlayer is turned on, the message "CARD INSERTED" will appear, and will then be followed by the Home Screen. This will indicate that the installation was successful and the card is functional. The MadPlayer is now ready to use.

Card Removal—*Important Notice!*

The MadPlayer MUST be in Home Mode, or turned off, before you remove a SmartMedia™ Card. If a card is removed during any other Mode, the card may be irreversibly damaged.

Error Messages:

The following error messages can be displayed after the card insertion:

ALIEN CARD: The card inserted is not formatted. Try to remove the card and re-insert it. The message may be due to an incorrect card insertion. Otherwise, follow the instructions in section "Format SmartMedia™Card" on page 50 to format the card.

INVALID CARD: The card inserted is a type not supported by the MadPlayer. If you think the card is of a valid type listed in paragraph "**Storage**

Choices" above, try to remove and re-insert it. The message may be due to an incorrect card insertion. Otherwise, use another card.

WORN-OUT CARD: The card inserted has too many invalid blocks. This condition can appear if the SmartMedia™Card is old. Try removing and re-inserting the card. The message may be due to an incorrect card insertion. Otherwise, use another card.

MAKING MUSIC WITH THE MADPLAYER™

Generactive Music Mode

To enter **Generactive Music Mode** press the ♫ ***e-DJ*** direct-access key. You can also use the ►***play*** key, if you are in Home Mode. The LCD will display "e.DJ" in the status line above a list of Music Styles. [See Figure 5: Music Style Selection Screen.]

Figure 5: Music Style Selection Screen

To **Select a Music Style** use Joystick up/down.
To **Play a MadSong** in your selected style, press ►***play.***
The MadPlayer now starts generating a completely new MadSong in the style of your choice. As the song begins, the **Music Highway** screen appears [See Figure 6.]

Figure 6: Music Highway Screen

Music Highway Overview

The Music Highway screen represents the six highway "lanes," or elements, of a MadSong with which the user can interact: the four instrument lanes—Drums, Bass, Riff and Lead; plus, the Microphone lane and the Sample lane.
[Note: Some lanes can play several different instruments in a song. For instance, there can be four different types of Lead instruments and composed patterns, and the display will show Lead1, Lead2, Lead3 and Lead4, each giving information about Lead instrument/pattern currently playing in the song.]

In Figure 6 the Riff Lane is the selected highway lane. The lane name appears at the bottom of the screen and a pulsing speaker hovers above the far end of the lane. (The speaker becomes larger or smaller with changes in volume, and disappears if the lane is muted.)

~~See Page 15 for Music Highway Controls.~~

Generactive Song Overview

Once a music style has been chosen and a song started, the MadPlayer will automatically introduce, build, climax and conclude the song, using the framework appropriate for that music style. If the user does not interact with the song, or change any other settings, a new song will immediately begin a few seconds after the conclusion of the previous song. (The length of the pause will depend upon whether the song that just finished was a slow or a fast song. Pauses will be slightly longer after a slow song than after a fast one.)

Tunnel Mode Overview

At any point during a MadSong the user can **Tunnel** beneath a Highway lane and change or modify that aspect of the song—Drums, Bass, Riff, Lead, Microphone or Sample. In Figure 7 the user has entered the Riff Lane Tunnel.

Figure 7: Riff Tunnel Screen

For as long as you remain in **Tunnel Mode**, the music sequence that was playing just before you **Tunneled** will continue to play as a two-bar loop, allowing you to interact with that music sequence until you are satisfied with the results. Depending on the Tunnel lane and music style, you can change the instrument that is playing, its music pattern, its speed, pitch, filter resonance, filter frequency, and more.

~~See page 16 for Tunnel Mode Controls.~~

MUSIC HIGHWAY CONTROLS

~~To interact with a song element, you must first select the Highway lane for that element—Drums, Bass, Riff, Lead, Microphone, or Sample. Although Tunnel Mode is where you can make the most extensive changes to elements of a song, don't overlook the controls that are available to you while you're on the Highway. ~~

- To **Select a Highway Lane** use Joystick left/right.
- To **Change Music Tempo** press the √ *Pitch/Tempo* key together with the Joystick. Use Joystick left to speed up the MadSong. Use Joystick right to slow it down.
- To **Change Music Pitch** press the √ *Pitch/Tempo* key together with the Joystick. Use Joystick up to raise the pitch. Use Joystick down to lower the pitch. (Note: If you adjust the music pitch while in the Mic Lane, you will change the pitch of the Mic input, and whatever sound/voice that is fed into the mic.)
- To **Pause the Music** press ▶*play.* PAUSED will flash in the status line.
- To **Resume the Music** press ▶*play* again.
- To **Mute the Lane Instrument** press ■ *stop*.
- To **Un-Mute the Lane Instrument** press ■ *stop* again.
- To **Solo a Lane Instrument** press ▶*play* for 2 seconds (Remember: A short press on ▶*play* will pause the music.). MadPlayer will play the instrument pattern continuously, while all other lanes are muted, until Solo Mode is exited by pressing ▶*play* again for 2 seconds. (If ▶*play* is pressed when an instrument is not playing, MadPlayer will compose the instrument pattern and play it starting at the end of the current 2-bar sequence.)
- To **Stop the Music** and return to the e-DJ Style Selection Screen, press ■ *stop* for 2 seconds.
- To **Start a New Song** press ▶▶ *forward. (Remember! You will lose the current song if you did not save it.)*
- To **Restart the Current Song** press ◀◀ *backward*.
- To **Change Lane Volume** press the ≡|~ *FX* key with Joystick up/down.
- To **Change Lane Reverb** press the ≡|~ *FX* key with Joystick left/right.
- To **Save the Current Song** press the ⊕ *Save/Edit* key. This will save it as a MadSong on the SmartMedia™ Card. A song can be saved at any time while it is playing. When it is stored, MadPlayer assigns a name to the song and displays it, so you can remember the generated name. Once it is stored on the card, you can re-name the song at any time. (See Renaming Song Files, page 22.)
- To **Play a Sample** press ▶*play* while you are in the Sample lane.
- To **Change the Sample Selection** press the ≡|~ *FX* key together with Joystick left/right, while you are in the Sample lane.

While on the Music Highway (and in tunnels), you can see the active Equalizer preset on the LCD screen by pressing the ≡|~ *FX* key. You also can change the active preset by holding down the ≡|~ *FX* key and simultaneously pressing the

♂ ***System*** key. If you want to see and/or edit the active preset , hold √ *Pitch/Tempo* key down while pressing ♂ ***System*** key. This will take you to the Equalizer menu (see page 51).
The music will keep playing while you do any of the above, and the effects you are adding will be applied in real time.

TUNNEL MODE CONTROLS

~~Many song elements can only be modified from within Tunnel Mode. Once you enter Tunnel Mode, the last two bars of the current MadSong will loop indefinitely, until you use the Joystick to re-enter Highway Mode.~~

- To **Enter Tunnel Mode** select a Highway lane while in Highway Mode. Then press Joystick down. You will now see a Tunnel Screen [Figure 7] with the Tunnel name in the status line.
- To **Return to Music Highway Mode** press Joystick up.

Instrument Tunnel Controls—Drums, Bass, Riff or Lead:

- To **Save the Current Music Pattern** for the lane you are in, and to **Compose a New Music Pattern** press Joystick right.
- To **Discard the Current Music Pattern** and **Compose a New Music Pattern** press Joystick down.
- To **Return to a Saved Music Pattern** press Joystick left. (See Music Pattern Numbering, Page 39, for more details about saving Music Patterns.)
- To **Solo an Instrument** press ▶ ***play.*** All other lanes will be muted.
- To **Un-Solo an Instrument** press ▶ ***play*** again. All other lanes will then be un-muted. [Note: if you Solo several lanes, you have to go back to each Lane and Un-Solo that lane instrument, if you want to again hear all the elements of the song.]
- To **Mute an Instrument** press ■ ***stop.***
- To **Un-Mute an Instrument** press ■ ***stop*** again.
- To **Change the Instrument** press ▶▶ ***forward*** or ◀◀ ***backward*** to cycle through all available instruments—if you are in the Riff Tunnel, for example, you will hear all instruments the MadPlayer makes available to play that riff. A "Changing instr." message will appear on the screen while MadPlayer composes the same music pattern with a different instrument. A small * at the end of the message indicates that MadPlayer™ has cycled through all compatible instruments. It will then replay the pattern using the original instrument (i.e. the one that was playing before you started the changes).
- To **Change Instrument Filter Frequency** press the ≡|- *FX* key together with Joystick up/down. (This is only available in Techno or Hip-Hop styles). The current filter value is displayed, with an * to its right, to indicate the initial value set before changes.
- To **Change Instrument Filter Resonance** press the ≡|- *FX* key together with Joystick left/right. (This is only available in Techno or Hip-Hop styles). The

current filter value is displayed, with an * to its right, to indicate the initial value set before changes.

~~Instruments that have been Muted or Soloed will remain that way when you return to the Music Highway. You must go back to the Muted/Soloed lanes, and Un-Mute or Un-Solo those instruments, if you want to again hear all elements of the song.~~

Sample Tunnel Controls:

- To **Select a Sample** press Joystick left/right. This will let you cycle through all available Samples for the current Music Style.
- To **Control Sample Volume** press the *FX* key together with Joystick up/down.
- To **Modify a Sample** press the *FX* key together with Joystick left/right. This lets you apply a sound effect to the sample—Low Voice, Doppler, Reverb, and more. [Note: Custom effects can be user-defined in the User Config command from the System menu (see page 51)].
- To **Play a Sample** press ▶ ***play***.
- To **Stop a Playing Sample** press ■ ***stop***.
- To **Mute the Sample Lane** press ■ ***stop*** once, if there is no sample currently playing; press ■ ***stop*** twice to mute the lane while a sample is already playing.
- To **Un-Mute the Sample Lane** press ▶ ***play***.

Microphone Tunnel Controls:

- To **Change Input Pitch** press the *Pitch/Tempo* key together with Joystick up/down.
- To **Create Echo Effects** press the *FX* key together with Joystick left/right.
- To **Control Microphone Volume** press the *FX* key together with Joystick up/down.
- To **Control Microphone Balance** press Joystick left/right.
- To **Mute Microphone** press ■ ***stop***.
- To **Un-Mute Microphone** press ▶***play***.

SONG STORAGE & PLAYBACK

~~MadPlayer™, equipped with a SmartMedia™ Card, can Save and Playback all the music you want to hear.~~

Song Storage Basics:

MadPlayer can save/play songs in a wide range of audio formats. Songs in all supported formats can be saved on the same SmartMedia™ Card—and even in the same Playlist. This version of MadPlayer supports the following song formats:

WMA (*aka* ASF)	Windows Media Audio files
MP3	When converted into WMA files (MadWare™ PC software does it for you!)
MIDI	Midi audio files
KARAOKE	Midi Karaoke songs
MADSONGS	Generactive MadPlayer songs

Song Formats are shortened to their first three letters (Karaoke is KAR, etc.) when MadPlayer lists the audio format of a song in its status bar.

- To **View the Song Menu** press the ◎ ***Songs*** direct access key. Using Joystick up/down, select PLAYLISTS (for songs collected under a Playlist title), or SONGS (for access to individual songs), and then press ▶***play***. What appears is a scrollable Directory of all Songs or Playlists that are stored on your current SmartMedia™ Card

In Figure 8, the Directory of Songs has been opened. The upper left corner of the status bar reads "e.songs," meaning that you have opened the Songs Directory. The upper right corner reads WMA, because the selected song is a WMA format audio file. The bottom song name is the selected title. If you now pressed ▶***play***, the WMA-format song would begin to play.

If, instead, the Playlist Directory had been opened, the upper left corner would read "Playlists," there would be no entry in the right corner, and the selected Playlist title would appear in the bottom box of the pyramid of titles. If you then pressed ▶***play*** the first song entry in the selected Playlist would begin to play.

Figure 8: Song Menu Directory

- To **Select a Song** use Joystick up/down.
- To **Start playing the selected song**, press ▶***play.***
- To **go to next page** (next group of 3 songs) press ▶▶***forward.***
- To **go to previous page** (previous group of 3 songs) press ◀◀***backward.***

Saving/Playing Your Generactive MadSongs

When you save a MadSong the MadPlayer automatically assigns a name to the song as it is stored on the SmartMedia™ Card. For example, if MadPlayer is playing a Techno style song the saved song will be named HIPHOP001, the next saved HIPHOP song would be HIPHOP002, etc. Each music style has its own prefix (such as HIPHOP) and the first saved song in that music style will be 001. If you delete a song—say

HIPHOP003—from your list of HIPHOP001 through 005, the next time you save a song in that style, MadPlayer will fill in the gap and name it HIPHOP003.

If you save a song which you've already saved once, say as HOUSE003, MadPlayer will save it as HOUSE003V01. If you work on the song a third time, it will be saved as the third version, HOUSE003V02, and so on. Your original version, and each version after, will play just the way you saved it.

After the song has been saved, you can rename it anything you want, and can also choose to copy it to a Playlist. (See Song Names and Playlists, Page 18.)

Unlike songs saved in the other supported formats, if you select a MadSong from the directory and press ▶*play*, MadPlayer will immediately revert to six-lane Music Highway Mode while the song is playing.

At the end of the selected song, unless you intervene, MadPlayer will automatically begin playing the next song that appears alphabetically in the Song Directory of the SmartMedia™ Card—even if the next song in the directory is not a MadSong.

~~*All controls for **Saving and Playing MadSongs** are included in **Music Highway Controls**, Page 12.*~~

Saving/Playing Non-MadPlayer Songs

MadPlayer can download and playback up to 120 minutes of non-generactive digital music. Songs, Playlists and Samples can all be stored on the same SmartMedia™ Card—using any one of, or all of, the supported audio formats. (See Supported Audio Formats, Page 40.) And all supported formats can even be combined within the same Playlist, along with your own Generactive MadSongs.

When a Non-MadPlayer song is selected and played, a version of the screen in Figure 9 is displayed. The status line begins with the song's file format, in this case WMA, next is the length of the song, in minutes and seconds, and on the right is the elapsed time since the song began playing. At the center of the screen the title of the song, Fragile, scrolls in a banner, for as long as the song is playing.

If the song is a Karaoke file, the lyrics are displayed on two lines at the bottom of the screen, and the animated frame is not displayed. (See KARAOKE MODE, Page 43.)

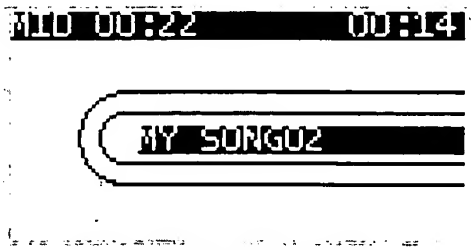


Figure 9: Song Play Screen

Non-MadPlayer Song Controls

- To **Play a Saved Non-MadPlayer Song** press the ◎ *Songs* direct access key, and then select "Songs" and press ▶*play*. Scroll through the list of songs on the current SmartMedia™ Card using Joystick up/down. When the correct song title is in the selection box press ▶*play*. .
- To **Edit a Saved Non-MadPlayer Song** see Song Editing, page 20, which explains all ways in which MadPlayer and Non-MadPlayer songs can be edited.
- To **Change Music Tempo** press √ *Pitch/Tempo* key together with Joystick right to speed up tempo, or Joystick left to slow down tempo.
- To **Change Music Pitch** press √ *Pitch/Tempo* key together with Joystick up to raise the pitch, or Joystick down to lower the pitch.
- To **Pause Playback** press ▶*play* once. "PAUSED" flashes in display area.
- To **Resume Playback** press ▶*play* again.
- To **Stop Playback** press ■ *stop*. This returns you to the current Song Directory.
- To **Skip to Next Song** press ▶▶*forward* briefly. If playing a Playlist this skips to the next song on the list. Otherwise, the command skips to the next alphabetical song in the SmartMedia™ Card Song Directory.
- To **Fast Forward A Song** press and hold ▶▶*forward.* The status bar timer continuously registers the elapsed time as you move through the song. If you Fast Forward past the current song, MadPlayer will stop approximately 10 seconds before the end of the song.
- To **Restart The Song** press ◀◀*backward* briefly. This will replay the song from the start.
- To **Return to Previous Song** press ◀◀*backward* briefly, twice (like a double click). If playing a Playlist, this will replay the previous song in the list. Otherwise, the command will replay the previous alphabetical song in the SmartMedia™ Card Song Directory.
- To **Reverse (Rewind) Through a Song** press and hold ◀◀*backward.* The status bar timer continuously registers the decreasing elapsed time as you Rewind through the song. If you Reverse beyond the current song, MadPlayer stops at the beginning of the song.

Downloading Songs from a PC

Please refer to the MadWare™ User's Guide, included on the MadWare™ CD.

EDITING SONG FILES

Once a song is created by the MadPlayer, or is imported into MadPlayer and stored on the SmartMedia™ Card, it can be edited or modified in a number of ways. The Song Edit screen allows you to make changes to as many as five different song elements. Some song elements can be changed through a number of different edit features of the MadPlayer, others can only be changed through this specific edit screen.

Opening the Song Edit Screen

- To **Open the Song Menu** press the ◎ *Songs* direct access key, select "Songs" and then press ▶ *play.*
- To **Open a Song for Editing** scroll through the Songs Directory, select the desired song, then press the ⇨ *Save/Edit* key. The Song Edit screen for the selected song is now displayed.

In Figure 10, the Song Edit screen for a MadSong called HipHop001 is pictured. "Edit HipHop001" appears in the status line. Below that, in the edit screen, is the list of song elements, on the left, and the editable parameters or attributes of those elements on the right.

Figure 10: Song Edit Screen

What can be edited?

For MadSongs the Name, Speed, Pitch, Duration and Sample Set can be edited.
For Non-MadSongs only the Name, Speed and Pitch can be edited.

Using the Song Edit Screen

Once the Song Edit screen for a song file is opened, use Joystick up/down to select the song element you want to change. Then use Joystick right to highlight the editable parameter/attribute of that song element. Now using Joystick up/down will cycle you through the choices that are available for that parameter/attribute. When have selected your choice, you can either save the one change and exit, exit without saving the change, or you can make changes to as many elements of the song as you want and then save all the changes when you exit—or exit without saving any of them.

- To **Select a Song Element** use Joystick up/down.
- To **Edit a Selected Song Element** use Joystick right to highlight the parameter/attribute to be edited.
- To **Scroll Through the Edit Choices** use Joystick up/down.
- To **Select the Edit Change and Jump** to the next line press ▶ *play.*
- To **Manually Move to Another Song Element** use Joystick left, then Joystick up/down.
- To **Save the Edit Change(s) and Exit** press the ⇨ *Save/Edit* key.
- To **Abort the Edit Change(s) and Exit**, press ■ *stop* twice, or press any direct access key.

Renaming Song Files

Use Joystick up/down to select NAME, then use Joystick right to select the current title of the song. Once selected, the first letter of the title will begin flashing. Follow the commands below to revise or change the song's title. The maximum song title length is 12 characters. (Renaming Song files can also be done through the Files Menu, see Renaming Songs, Playlists, Samples and Sample Sets, pages 22 and following.)

- To **Erase the Title** press ■ ***stop***. A blinking 'A' will appear on the left. Follow the procedures below to insert/change characters of the title.
- To **Select a Title Character** to be changed, use Joystick left/right. The selected character will blink.
- To **Change a Selected Character** use Joystick up/down to cycle through the letters of the alphabet. When correct character is selected move to next character to be changed, or Save the new title.
- To **Insert an Extra Character** to the left of a selected character, press ▶▶***forward.***
- To **Add a Character** at the end of the title, select the last character and press **Joystick right.**
- To **Delete a Character** first select it, then press ◀◀ ***backward***.
- To **Save the New Song Title** press the ⮊ *Save/Edit* key. This will save the new title and return you to the Songs Directory.
- To **Quit Without Saving Song Title Changes** press ■ ***stop*** three times. This returns you to the Songs Directory. Or press any direct access key to leave the Song Edit screen without saving changes.

Changing Speed & Pitch

Both Speed and Pitch can be adjusted while a song is actually playing, by using the Music Highway Controls, page 15. Although you can also change both aspects of the song from within the Song Edit screen, you cannot listen to the song in this mode, so you cannot hear what the adjustments will do to it.

But, if you need to make a specific Speed or Pitch adjustment to an existing song, and you know the exact number of the setting that you want, then using this screen to make the change(s) can be easy and efficient.

First, use Joystick up/down to select either Speed or Pitch, then use Joystick right to select the numeric setting for that element. Use Joystick up/down to cycle through the available numeric settings. When the desired Speed or Pitch number is selected, press the ⮊ *Save/Edit* key to save the change and exit.

Modifying Song Length

Duration, the length of a song, can only be modified using the Song Edit screen. Although the length of the song—in minutes and seconds—can be changed here, you will have to return to the Song Directory, select the song, and press ▶***play*** in order to hear the new longer or shorter version of the song. The maximum length of a MadSong is 10

minutes, and the minimum is 1 minute. Only MadSongs can be modified in this way. You cannot increase or decrease the length of a Non-MadPlayer song.

First, use Joystick up/down to select DURATION, then use Joystick right to select the Minute:Second numeric length for the current song. Use Joystick up/down to cycle through all the available lengths for the song. When the desired length is selected, press the ⮊ *Save/Edit* key to save the change and exit.

Associating Sample Sets With Songs

After you have created a Sample Set (See Creating Sample Sets, page 32.) you can associate that set with a particular song. That means that when you play the song, only the samples that are part of the associated set will be listed in the Samples Lane, and, if you have Auto Sample turned on (See Auto Sample in System Configuration, page 46) only samples from the associated sample list will automatically be inserted into a song by MadPlayer. (You can also associate Sample Sets with an entire Song Style, see How to Associate Samples With Specific Song Styles, page 23.)

To choose a Sample Set, first use Joystick up/down to select SAMP SET. Then use Joystick right to select the current association—either "None," or a sample set name, such as SMPSET001. Now use Joystick up/down to cycle through the available Sample Sets. (There won't be any if you haven't created them.)

When the desired Sample Set is selected, press the ⮊ *Save/Edit* key to save the change and exit.

{Is this all correct about when samples are available from an associated set??? I know it's not finalized in the MadPlayer I'm working with.}

*~~See **Managing Music Files, page 36** , to learn how to Copy Songs, Delete Songs, or Rename Songs, using a different Edit Screen.~~*

PLAYLISTS

What are Playlists?

A Playlist is a collection of songs that are grouped together under a single Playlist title. When you select and play that Playlist title, all the songs in the collection will be played one after the other. See Playing Playlists, page 26.

Any song stored on the SmartMedia™ Card can be added to a Playlist. And songs of all supported formats can be collected together in the same Playlist.

Creating Playlists

- To **Create a Playlist** press the ◎ *Songs* direct access key, then select PLAYLISTS and press ▶ *play* to display the existing Playlists (if any). Now press ▶▶ *forward* to create a new Playlist.

The name of a new Playlist will appear in the status bar at the top of the screen. If this is the first Playlist created, MadPlayer will call it PLAYLIST001. The next created Playlist will be called PLAYLIST002, etc.

When a Playlist is first created it contains only one song, and the termination instruction for the list. The song that is automatically inserted into the Playlist is the alphabetically first song on the SmartMedia™ Card. You can keep the song in the list, or remove it, the song is only there to create the first song slot in the list. Once the first slot is created you can add, remove, rename and change the order of as many other songs as you wish to store in that list.

The last line of the Playlist contains the termination instruction, which tells MadPlayer to either End List or Loop List. The End List instruction means that MadPlayer will stop playing selections from the Playlist after it plays the last song in the list. Loop List means that MadPlayer will begin re-playing the whole list, from the top, after it has played the last song selection of the list. When a new Playlist is created the default termination instruction is End List. But you can change the termination choice—using Joystick up/down—when you add songs to the Playlist.

Editing Playlists

Adding, deleting, and changing the order of songs in a Playlist are all accomplished through the Playlist Edit Screen. (The actual title of the Playlist can also be changed within the Edit screen.) Below are the steps to open the Playlist Edit Screen, followed by step-by-step instructions.

Opening Playlist Edit Screen

- To **Open a Playlist for Editing** press the ◎ ***Songs*** direct access key, then select PLAYLISTS and press ▶***play***. Scroll through the Playlist Directory using Joystick up/down until the correct Playlist title is selected. Then press the ⮆ *Save/Edit* key. The Edit screen for the Playlist now appears.

Figure 11, below, shows the Edit screen for a Playlist. The Playlist title appears in the status line. The first line below the status line displays "Name" in the left column and the current Playlist name to the right. Beneath that is the list of songs in the Playlist—their numerical order in the left column, their title in the right column. The last line of the list contains the termination instruction—End List.

Figure 11: Playlist Edit Screen

Using the Playlist Edit Screen

Once the Playlist Edit Screen is opened, use Joystick up/down to select the element you want to change. Then use Joystick right to highlight the editable parameter or attribute. Now using Joystick up/down will cycle you through the choices that are available for that parameter/attribute. When have selected your choice, you can either save the one change and exit, exit without saving the change, or you can make changes to as many elements as you want and then save all the changes when you exit—or exit without saving any of them.

- To **Select an Entry** use Joystick up/down.
- To **Edit a Selected Entry** use Joystick right to highlight the parameter/attribute to be edited.
- To **Scroll Through the Edit Choices** use Joystick up/down.
- To **Select the Edit Change and Jump** to the next line press ▶ ***play.***
- To **Manually Move to Another Entry** use Joystick left, then Joystick up/down.
- To **Save the Edit Change(s) and Exit** press the ⇳ *Save/Edit* key.
- To **Abort the Edit Change(s) and Exit**, press ■ ***stop*** twice, or press any direct access key.

Renaming Playlists

Use Joystick up/down to select NAME, then use Joystick right to select the current title of the Playlist. Once selected, the first letter of the title will begin flashing. Follow the commands below to revise or change the Playlist title. The maximum title length is 12 characters. (Renaming Playlists can also be done through the Files Menu, see Renaming Songs, Playlists, Samples and Sample Sets, pages 22 and following.)

- To **Erase the Title** press ■ ***stop***. A blinking 'A' will appear on the left. Follow the procedures below to insert/change characters of the title.
- To **Select a Title Character** to be changed, use Joystick left/right. The selected character will blink.
- To **Change a Selected Character** use Joystick up/down to cycle through the letters of the alphabet. When correct character is selected move to next character to be changed, or Save the new title.
- To **Insert an Extra Character** to the left of a selected character, press ▶▶ ***forward.***
- To **Add a Character** at the end of the title, select the last character and press **Joystick right.**
- To **Delete a Character** first select it, then press ◀◀ ***backward.***
- To **Save the New Playlist Title** press the ⇳ *Save/Edit* key. This will save the new title and return you to the Playlist Directory.
- To **Quit Without Saving Playlist Title Changes** press ■ ***stop*** three times. This returns you to the Playlist Directory. Or press any direct access key to leave the Playlist Edit screen without saving changes.

Adding a Song to a Playlist

Use Joystick up/down to highlight the number of the Song below which you want to insert a new Song. Now press ▶▶ *forward.* A new line appears below the line you highlighted. On the left it contains the next number in the Playlist sequence, on the right the next Song that appears alphabetically on the SmartMedia™ Card has been inserted in the Playlist. (If you want to add 10 Songs to the list, repeat this procedure until MadPlayer has inserted 10 Songs that are chosen alphabetically from the SmartMedia™ Card. Then you can follow the instructions below to change those Song selections into the ones you want to include in the list.)

Changing a Playlist Song Selection

First use Joystick up/down to highlight the number of the song you want to change. Then use Joystick right to highlight the Song Title. Now use Joystick up/down to browse through the list of songs that are stored on the SmartMedia™ Card. When the song you want to use as a replacement is selected, either Save your changes, or move to another line of the Edit Screen to continue editing.

Deleting a Playlist Song Selection

First use Joystick up/down to highlight the number of the Song you want to delete. Now press ◀◀ *backward.* The Song Title is then deleted from the Playlist. (Note: Deleting a song from a Playlist does not delete the original song file from the SmartMedia™ Card.)

Changing the Order of Songs in a Playlist

First delete the song from the list using Deleting a Playlist Song Selection above. Then insert the song elsewhere in the list using Adding a Song to a Playlist above.

Changing Playlist Termination Instruction

First use Joystick up/down to select the line of the Termination Instruction. Then use Joystick right to highlight the actual instruction—End List or Loop List. Use Joystick up/down to change the instruction.

Playing Playlists

- To **Play a Playlist** press the ◎ *Songs* direct access key, then select PLAYLISTS and press ▶ *play*. Scroll through the list of Playlists on the current SmartMedia™ Card using Joystick up/down. When the correct Playlist is in the selection box press ▶ *play*. The first song in the Playlist collection will begin to play.

If the terminating instruction of the Playlist is End List, then at the end of the last song MadPlayer will revert to the Songs Directory. If the terminating instruction is Loop List, then MadPlayer will begin replaying the whole collection of Playlist songs, first to last, as soon as the last song ends.

*~~See **Managing Music Files, page 36** , to learn how to Copy Playlists, Delete Playlists, or Rename Playlists, using a different Edit Screen.~~*

SAMPLING

About Samples and Sampling

With MadPlayer™ you can record sound samples from almost any source—songs you've saved on the SmartMedia™ Card, the MadPlayer FM Radio, your own voice, and anything else that you can record with the MadPlayer Microphone. Samples can also be transferred from a PC, where the samples have already been pre-recorded. Such samples may have been downloaded from a web site, created with a wave editor PC program, or brought into existence in some way still to be discovered.

When recording via the MadPlayer, the recording format is the one defined in the System Configuration menu (see System Configuration: REC FORMAT parameter, Page 46) which can be changed by the user through the same menu (System Configuration, Page 46).

All samples can be modified once they are saved, allowing you to change the sample name, add sound effects, adjust relative volume, and much more.

CREATING/RECORDING SAMPLES USING THE MADPLAYER

Samples can be recorded in many modes of MadPlayer operation, but not when playing WMA music files. Once you begin recording a sample, recording automatically stops if the sample file exceeds 500Kbytes—approximately 45 seconds. The MadPlayer automatically chooses the recording source, or mix of sources, depending on the circumstances.

[The sampling frequency and the compression, which both determine the sound quality and the sample size, can be determined in the Config System menu. Please refer to System Configuration on page 46 for more details.]

- In the **Samples Mode** (You are in this mode when you have pressed the ***Samples*** direct access key) the Microphone input is the recording source.
- When **playing a MadSong, a MIDI or a KARAOKE song**, the recording source is the Mix between the music playing and the Microphone input.
- When **listening to the radio**, the recording source is the radio.
- In **all other modes**, the recording function is disabled. If you press the ●***record*** key, the message "CANNOT RECORD" is displayed.

- To **Record a Microphone Sample**, press the ***Samples*** direct access key to enter the Samples mode. Then press ●***record*** to start recording. Press the ●***record*** key again to stop recording the sample. (If you wait longer than 45 seconds the recording will automatically be concluded.). The status line displays "Recording SAMPLExxx" while you are recording.

- To **Record a Music Sample,** meaning music being played by the MadPlayer, first mute the microphone by pressing √ *Pitch/Tempo* together with Volume down—if there is other noise in the room that you don't want picked up by the microphone. Select and play the song as described in SONG STORAGE & PLAYBACK, Page 17. Then, when the portion of the song you want to sample begins, press ●***record.***
  Press ●***record*** a second time to stop recording the sample, or let the recording stop automatically after approximately 45 seconds.

  Music Sampling Note: the only types of songs that cannot be sampled are WMA songs. Also, if you sample a song that already contains samples you have created, those samples will not be recorded on the new sample.

- To **Record a "Microphone + Music" Sample,** first select and play the song, as described in SONG STORAGE & PLAYBACK, Page 17. Then, when the portion of the song you want to sample begins, press ●***record.*** MadPlayer will now record the song along with whatever you supply for Microphone input—your voice, your dog barking, your favorite bongo riff, etc.
  Press ●***record*** again to stop recording the sample, or let the recording stop automatically after approximately 45 seconds.

Microphone Controls

Control shortcuts available in all modes

- To **Mute the Microphone** press √ *Pitch/Tempo* key together with Volume down key.
- To **Un-mute the Microphone** press √ *Pitch/Tempo* key together with Volume up key.
- To **Decrease the Microphone Volume** press ≡|- *FX* key with Volume down key.
- To **Increase the Microphone Volume** press ≡|- *FX* key together with Volume up key.

Other Controls:

- To **Change Pitch of Microphone Input** press √ *Pitch/Tempo* key together with Joystick up to raise the pitch, Joystick down to lower the pitch.
  (This control is available in all modes except the following: Home Screen, Music Highway--other than Microphone lane, Tunnel Mode--other than Microphone tunnel, while playing WMA songs or samples, and in System Menu.)

- To **Modify Microphone Echo Level** press ≡|- *FX* key together with Joystick left/right.

(This control is available in all modes except the following: Home Screen, Music Highway--other than Microphone lane, Tunnel Mode--other than Microphone tunnel, while playing WMA songs or samples, and in System Menu.)

- To **Increase/Decrease Microphone Volume** press ≡|- *FX* key together with Joystick up/down.
  (This control is available in all modes except the following: Home Screen, Music Highway--other than Microphone lane, Tunnel Mode--other than Microphone tunnel, while playing WMA songs or samples, and in System Menu.)

Sample Storage

After the sample is recorded MadPlayer will save it and automatically assign a name to the sample. That name will be displayed in the status bar. The first sample you save will be called SAMPLE001, the next will be SAMPLE002, etc. If you record eight samples (SAMPLE001—SAMPLE008) and then delete SAMPLE005, the next sample you record will be assigned the name SAMPLE005—because it would again be the first available number for a new sample. And the next sample after that would be assigned SAMPLE009—because then 009 would be the first available number for a new sample. Once the sample has been saved, the user will be returned to the Samples menu.

After the sample is saved you can, if you want to, change the assigned sample name through the RENAME function that can be accessed through two menus: See Renaming Songs, Playlists, Samples and Sample Sets, pages 22 and following; or, refer to Renaming Samples and Sample Sets, pages 30 and following.

Modifying Sample Effects & Volume

After a Sample has been saved, the Sample Menu allows you to listen to the Sample and experiment with possible Sound Effects and Volume adjustments.

- To **Enter the Samples Menu** press the 🖹 ***Samples*** direct access key. Select SAMPLE SETS for access to collections of samples, or SAMPLES for access to individual sample files. Then and press ▶***play***.

If SAMPLES was selected, "e.Samples" is displayed in the left side of the status line, the right side displays the effect—such as DOPPLER—that has been applied to the Sample that appears in the bottom selection box.
If SAMPLE SETS was selected, "Sample Set" appears in the status line.

- To **Play to a Sample**, use Joystick up/down to select the title, then press ▶***play.***
- To **Play All Samples in a Sample Set**, use Joystick up/down to select the Sample Set title, then press ▶***play.***

You can select a sound effect to be applied to a sample and then listen to the result, all within this menu. You can also adjust the sample volume, to control how loud the sample will be within the context of a song. If you want to change the volume of the sample permanently, use the SAMPLE GAIN parameter in the Sample edit screen (see below).

- To **Apply a Sound Effect**, first select the sample title—but do not press ▶*play.* Then press the *FX* key together with Joystick left/right. The available effects—Doppler, Reverb, Wobbler, Custom, etc—will be displayed in the right side of the status line. To hear the sample with the currently displayed effect, press ▶*play.* [Note: In order to PERMANENTLY apply a sound effect to a given sample, you must use the SAMPLE edit screen (see page 31).]
- To **Change Sample Volume** press the *FX* key together with Joystick up/down.

~~If you now want to assign a specific Sound Effect to a Sample, see Editing Samples, page 30 , for step-by-step instructions.~~

EDITING SAMPLES

Once a sample has been recorded and named you can open the sample with an edit menu and modify certain elements.

- To **Open a Sample for Editing** press the ***Samples*** direct access key, then select SAMPLES and press ▶***play***. Scroll through the list of Samples using Joystick up/down until the desired Sample is selected. Then press the *Save/Edit* key. The Sample Edit screen now appears.

Figure 12 displays the Sample Edit screen for the sample called SAMPLE001. The status line reads Edit SAMPLE001. Below it are the Sample Elements. The first five entries of the list can be edited: Name, Def Effect, Effect Type, Sample Gain and Sample Type. Other entries are here for information only and cannot be edited.

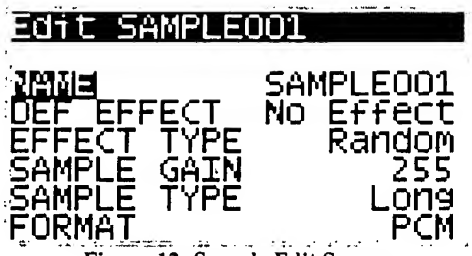


Figure 12: Sample Edit Screen

Renaming Samples

- To **Rename a Sample** use Joystick up/down to highlight NAME, then use Joystick right to highlight the actual Sample title. The first letter of the title will now begin flashing.

You can now change any letter of the Sample title, or add/delete letters, to create a new Sample title. The maximum title length is 12 characters. Just follow these steps:

- To **Erase the Title** press ■ ***stop***. A blinking 'A' will appear on the left. Follow the procedures below to insert/change characters of the title.
- To **Select a Title Character** to be changed, use Joystick left/right. The selected character will blink.
- To **Change a Selected Character** use Joystick up/down to cycle through the letters of the alphabet. When correct character is selected move to next character to be changed, or Save the new title.
- To **Insert an Extra Character** to the left of a selected character, press ▶▶ ***forward.***
- To **Add a Character** at the end of the title, select the last character and press **Joystick right.**
- To **Delete a Character** first select it, then press ◀◀ ***backward.***
- To **Save the New Title** press ⮊ *Save/Edit* key. This will save the new title and return you to the Samples Menu.
- To **Quit Without Saving Title Changes** press ■ ***stop*** three times. This returns you to the Samples Menu. Or press any other direct access key to Quit without saving title changes.

Important Note:
After you rename a Sample you must also re-insert that file in any Sample Set in which the old name is listed, otherwise the Sample will not play when you use those Sample Sets. To do this, follow the instructions in Modifying Sample Sets, page 32, to insert the new name. If you try to play a Sample Set from which some of the sample names have been removed/deleted, a "Lost Item" message will appear in the status bar.

Default Effect

This allows you to choose the effect that will be used on the Sample. Use Joystick up/down to highlight DEF EFFECT. Then use Joystick right to highlight the current effect: No Effect, Reverb, Wobbler, Doppler, Hi Voice, Lo Voice, or any user-defined custom effect. Now use Joystick up/down to cycle through the possible effects. Save changes as described above.

Effect Type

This allows you to choose whether a single effect (the default effect, see above) will be used on the Sample, or whether MadPlayer can employ random effects when playing the sample in an e-DJ song (or a MadSong). [Note: If random is selected, the first time the sample is played, it will be played with Default effect. Subsequent plays will randomly use the other effects.]

Sample Gain

This allows you to adjust the Gain numerically, by scrolling up or down with the Joystick. This feature permits you to permanently adjust the volume level at which the sample will be played later on, in case it was recorded too soft or too loud.

Sample Type

This allows you to choose whether the sample will be a Long or Short sample. Sample types are used by the MadPlayer when automatically playing samples (AUTOSAMPLE feature must be ON). Typically, Short samples are used by the MadPlayer algorithms to mark beats while Long samples are used as recurring "phrases". By default, if the sample is 1 second long or less, the sample type will be Short. If the sample is longer than 1 second, the sample type will be Long.

CREATING SAMPLE SETS

- To **Create a Sample Set** press the *Samples* direct access key, then select SAMPLE SETS and press ▶*play* to open the current SmartMedia™ card and display the existing Sample Sets (if any). Now press ▶▶*forward.* A new Sample Set is now created.

The name of the new Sample Set will now appear in the status bar at the top of the screen, after the word "New." If this is the first Sample Set created, MadPlayer will call it SMPSET001. The next created Sample Set will be called SMPSET002, etc.

When a Sample Set is first created it contains only one sample. The sample that is automatically inserted into the Sample Set is the alphabetically first sample on the SmartMedia™ Card. You can keep the sample in the list, or remove it, the sample is only there to create the first sample slot in the list. Once the first slot is created you can add, remove, and change the order of as many other samples as you wish to store in the Sample Set.

Modifying Sample Sets

Adding, deleting, and changing the order of Samples in Sample Sets (as well as Renaming Sample Sets), are all accomplished through the Sample Set Edit Screen. Below are the steps to open the Sample Set Edit Screen, and to then carry out the following actions.

Sample Set Edit Screen

- To **Open a Sample Set for Editing** press the *Samples* direct access key, then select SAMPLE SETS and press ▶*play*. Scroll through the Sample Set Directory using Joystick up/down until the correct Sample Set title is selected. Then press the *Save/Edit* key. The edit screen for that Sample Set now appears.

Figure 13, below, shows the edit screen for a Sample Set titled SamSet001. The status line reads, "Edit SMPSET001." The left column of the edit screen contains the NAME, STYLE, and numerical order of the Samples included in the Sample Set, and an End List instruction. All of these entries can be edited.

Figure 13: Sample Set Edit Screen

Using the Sample Set Edit Screen

Once the Sample Set Edit screen is opened, use Joystick up/down to select the element you want to change. Then use Joystick right to highlight the editable parameter or attribute. Now using Joystick up/down will cycle you through the choices that are available for that parameter/attribute. When have selected your choice, you can either save the one change and exit, exit without saving the change, or you can make changes to as many elements as you want and then save all the changes when you exit—or exit without saving any of them.

- To **Select an Entry** use Joystick up/down.
- To **Edit a Selected Entry** use Joystick right to highlight the parameter/attribute to be edited.
- To **Scroll Through the Edit Choices** use Joystick up/down.
- To **Select the Edit Change and Jump** to the next line press ▶ ***play.***
- To **Manually Move to Another Entry** use Joystick left, then Joystick up/down.
- To **Save the Edit Change(s) and Exit** press the ⮆ *Save/Edit* key.
- To **Abort the Edit Change(s) and Exit**, press ■ ***stop*** twice, or press any direct access key.

Renaming Sample Sets

Use Joystick up/down to select NAME, then use Joystick right to select the current title of the Sample Set. Once selected, the first letter of the title will begin flashing. Follow the commands below to revise or change the Sample Set title. The maximum title length is 12 characters. (Renaming Sample Sets can also be done through the Files Menu, see Renaming Songs, Playlists, Samples and Sample Sets, pages 22and following.)

- To **Erase the Title** press ■ ***stop***. A blinking 'A' will appear on the left. Follow the procedures below to insert/change characters of the title.
- To **Select a Title Character** to be changed, use Joystick left/right. The selected character will blink.
- To **Change a Selected Character** use Joystick up/down to cycle through the letters of the alphabet. When correct character is selected move to next character to be changed, or Save the new title.
- To **Insert an Extra Character** to the left of a selected character, press ▶▶ ***forward.***

- To **Add a Character** at the end of the title, select the last character and press **Joystick right.**
- To **Delete a Character** first select it, then press ◀◀ ***backward.***
- To **Save the New Sample Set Title** press the ⮆ *Save/Edit* key. This will save the new title and return you to the Samples Directory.
- To **Quit Without Saving Sample Set Title Changes** press ■ ***stop*** three times. This returns you to the Samples Directory. Or press any direct access key to leave the Sample Set Edit screen without saving changes.

Changing Sample Set Style Associations

Sample Sets can be associated with one or more Music Style, through the STYLE entry of the Sample Set Edit Screen. When a Sample Set is associated with a Music Style, the ▶***play*** will insert samples from that associated Sample Set when it is composing a new song in that Music Style, or when it is replaying a song in that Music Style and the Auto Sample function is turned on. (To learn more about Music Styles see Music Styles and Sub-Styles, Page 57.)

As well as being able to associate a Music Style with a Sample Set, you can also associate a Sample Set to a specific song. To associate a song with a Sample Set, see Associating Sample Sets With Songs, Page 23.

- To **Associate a Sample Set with a Style**, use Joystick right to highlight current Style choice. Then use Joystick up/down to cycle through the Style options.

Adding a Sample to a Sample Set

Use Joystick up/down to highlight the number of the Sample below which you want to insert a new sample. Now press ▶▶***forward.*** A new line appears below the line you highlighted. On the left it contains the next number in the Sample Set sequence, on the right the next Sample that appears alphabetically on the SmartMedia™ Card has been inserted in the Sample Set. (If you want to add 10 Samples to the list, repeat this procedure until MadPlayer has inserted 10 Samples that are chosen alphabetically from the SmartMedia™ Card. Then you can follow the instructions below to change those Sample selections into the ones you want to include in the list.)

Changing a Sample Selection in a Sample Set

First use Joystick up/down to highlight the number of the Sample you want to change. Then use Joystick right to highlight the Sample Title. Now use Joystick up/down to browse the list of Samples that are stored on the SmartMedia™ Card. When the Sample you want is selected, either Save your changes, or move to another line of the Edit Screen to continue editing.

Deleting a Sample from a Sample Set

First use Joystick up/down to highlight the number of the Sample you want to delete from the Sample Set. Now press ◀◀ ***backward.*** The Sample Title is then deleted from the

Sample Set. (Note: Deleting a Sample from a Sample Set does not delete the original Sample file from the SmartMedia™ Card.)

Changing the Order of Samples in a Sample Set

First Delete the Sample from the Set using Deleting a Sample from a Sample Set above. Then insert the Sample elsewhere in the Set using Add a Sample to a Sample Set above.

*~~See MANAGING MUSIC FILES, **page 36** , to learn how to Copy Samples or Sample Sets, Delete Samples or Sample Sets, or Rename Samples or Sample Sets, using a different Edit Screen.~~*

MANAGING MUSIC FILES

The Rename, Copy and Delete file functions for Songs, Playlists, Samples and Sample Sets are all available through the System Menu. The opening steps are the same to access all three functions. Those steps will be explained, below, and then each of the three tasks will be outlined in step-by-step explanations.

Opening the Files Menu

- To **Open System Menu** press the ☍ ***System*** direct access key.
- To **Select the Files Menu** use Joystick up/down to scroll through the list of sub-menus until Files is selected. Now press ▶***play.***

Figure 14 shows the opening screen of the Files Menu. The number of files present on the SmartMedia™Card appears on the left side of the status line. The right side indicates the amount of used memory. Below the status line is a separate line listing the amount of free memory left on the card, followed by a list of all the files that are stored on the card.

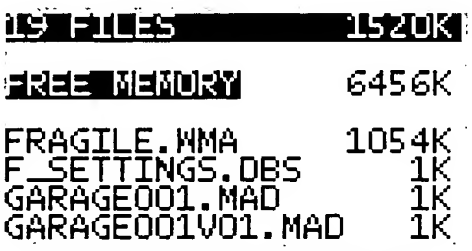


Figure 14: Files Menu Opening Screen

- To **Browse File List** use Joystick up/down.
- To **Scroll to Next Page** press ▶▶ ***forward.***
- To **Scroll to Previous Page** press ◀◀ ***backward.***
- To **Jump to Top of List** press Joystick left.
- To **Select a File** stop scrolling when filename is in flashing Reverse Highlight.
- To **Enter File Management** menu for the selected file, press ▶ ***play.***
- To **Exit the System Menu** without making changes, press ■ ***stop.*** Or press any Direct Access key to switch to another Mode.

Figure 15 shows the File Management Menu for the song file FRAGILE.WMA, which appears in the status line of the screen. Below it are the menu options— Delete, Rename, Attributes and Copy. (The Change Attributes function is discussed elsewhere. See Change Attributes Function, page 45.)

Figure 15: File Management Menu

- To **Rename, Copy or Delete a File**, use Joystick up/down to scroll through the list of options until Rename, Copy or Delete is selected—then press ▶*play.* Now follow the Rename, Copy or Delete file instructions, below.

Renaming Songs, Playlists, Samples and Sample Sets

Once you have selected Rename from the File Management options and pressed ▶*play,* the edit screen for the Song, Playlist, Sample or Sample Set filename will appear. Figure 16 displays the edit screen for the song titled Fragile. The status line reads "Rename File" and below it is the song title FRAGILE, with its first letter flashing, and ready to edit.

Figure 16: Rename File Edit Screen

You can now change any letter of the music filename, or add/delete letters, to create a new music filename for your Song, Playlist, Sample or Sample Set. The maximum music filename length is 12 characters. Just follow these steps:

- To **Erase the Title** press ■ *stop*. A blinking 'A' will appear on the left. Follow the procedures below to insert/change characters of the title.
- To **Select a Title Character** to be changed, use Joystick left/right. The selected character will blink.
- To **Change a Selected Character** use Joystick up/down to cycle through the letters of the alphabet. When correct character is selected move to next character to be changed, or Save the new title.
- To **Insert an Extra Character** to the left of a selected character, press ▶▶*forward.*
- To **Add a Character** at the end of the title, select the last character and press **Joystick right.**
- To **Delete a Character** first select it, then press ◀◀ *backward.*

- To **Save the New Music Filename** press ▶*play* or the ⮊ *Save/Edit* key. This will save the new title and return you to the File Menu.
- To **Quit Without Saving Filename Changes** press ■ *stop* twice. This returns you to the File Menu.
- To **Exit the File Menu** press any direct access key.

Important Note:
After you rename a Song you must also re-insert that file in any PlayList in which the old name is listed, otherwise the Song will not play when you use those PlayLists. To do this, follow the instructions in Editing PlayLists, Page 24, to insert the new name. If you try to play a PlayList for which some of the song names no longer exist, a "Lost Item" message will appear in the status bar.

Copying Songs, Playlists, Samples and Sample Sets

MadPlayer allows you to copy individual Songs or Samples, as well as entire Playlists or Sample Sets to either a new filename on the same SmartMedia™ Card, or to another SmartMedia™ Card altogether. This allows you to share songs and samples with friends, and it also allows you to create Playlists or Sample Sets that can exist in unlimited versions, for whatever purpose you have in mind.

Once you have selected Copy from the File Management options and pressed ▶***play,*** the edit screen for the Song, Playlist, Sample or Sample Set filename will appear. The first edit screen says "Destination Card" in the status line, and has two selectable choices: Other Card or Same Card.

Other Card:
You can transfer items from one card to another, as long as the file size does not exceed 16 KBytes. Typically, this will be sufficient to transfer MadSongs, Sample Sets, and Playlists, but not WMA songs, samples, etc.
After you select this choice and press ▶***play,*** the next screen says "Copy to other Card" in the status line, and a scrolling message tells you to "Remove Source Card". The MadPlayer has entered a safe mode and you can now remove the SmartMedia™ Card. Once the card is removed the message tells you to "Insert Destination Card." Take the other SmartMedia™ Card onto which you want to copy and insert it in the slot. Next MadPlayer says "Card Inserted." In the status line it now reads "COPY" along with the filename from which you are making a copy. And in the edit screen it says "Confirm," with a flashing YES. If you still want to copy the music file onto the card press ▶***play.*** If you wish to abort the process, use Joystick up/down to change Yes to No, and then press ▶***play*** .

If you said YES to Confirm, the Status line reads "Saving" and then reverts to the Home Screen. If you chose No, it aborts the process and then reverts to Home Screen. (Once MadPlayer is in Home Screen Mode, you can safely remove the SmartMedia™ Card and insert another card, if that is what you want to do.)

Same Card:

After you select this choice press ▶***play,*** the next screen says "Dest File Name" in the Status Line. This refers to the name of the destination file to which you will copy the original music file. In the edit screen below "Dest File Name," MadPlayer has suggested a filename, and the first letter of the name is flashing, so it is ready to edit—if you want to change the name by using the same set of commands that are listed in Renaming Songs, Playlists, Samples and Sample Sets, pages 22 and following.

The filename that MadPlayer suggests for a destination file is the next free filename available in that style—or of that type (SONG, SAMPLE, SAMPLESET, etc). For example, if you are copying Bossa002 to the Same Card. If Bossa003 and Bossa004 already exist, MadPlayer will suggest you name the new destination file Bossa005. You can accept that name or make your own filename. When you're satisfied with the name, press ▶***play.*** The next screen says "COPY, along with the filename you're copying from" and in the edit screen it says CONFIRM, with a flashing YES. Press ▶***play*** to OK the making of the copy. Change the YES to NO with the Joystick up/down and then press ▶***play,*** to abort the copy process. If you chose YES, "Saved" flashes in the status line and then the MadPlayer reverts to the song directory. The saved song is now in the directory, in alphabetical order.

Deleting Songs, Playlists, Samples or Sample Sets

After you select this choice and press ▶***play,*** the Status Line of the next screen says "Delete" along with the filename that is to be deleted. In the edit screen below it says CONFIRM, with a flashing YES. Press ▶***play*** to OK the deletion of the file. Change YES to NO with Joystick up/down and then press ▶***play*** to abort the deletion process. If you chose YES, "Deleting..." flashes in the Status Line, and then the screen reverts to the Song Directory. The file is now erased from the SmartMedia™ Card.

FM RADIO

- To **Enter Radio Mode** press the "A" ***Radio*** direct access key.

The LCD will display "v.Radio" in the status, along with the currently tuned radio frequency. The v.Radio screen lists the radio presets currently stored on the SmartMedia card. If no preset is present, a temporary message is displayed inviting you to press the ▶***play*** key to start listening to the radio.

Each station preset is assigned a name by MadPlayer. The first preset is named RADIO 01, the second is RADIO 02, etc. You can create as many as 20 station presets. Use the following commands to scroll through, select, and play your radio station presets:

- To **Select a Preset Station** use Joystick up/down.
- To **Listen to the Selected Preset Station** press ▶***play***.
- To **Create a new preset** press ▶▶***forward*** .
- To **Delete a preset** press ◀◀ ***backward***.
- To **Edit a Station Preset** press the ⮊ *Save/Edit* key. See "Editing radio presets," below for instructions on editing presets.

In Figure 17, the user has chosen to play preset RADIO01. The status line indicates "St" on the left side, if the signal received is Stereo, or "Mo" if the signal is mono, followed by "TUNED" if the reception level is strong, or "UNTUNED" if the reception level is weak. The radio frequency is shown on the right side. The name of the preset, RADIO 01, scrolls in the banner in the center-right of the screen while the station plays.

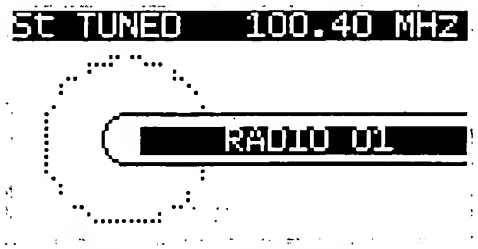


Figure 17: Play Radio Screen

If you want to tune to a new radio station, follow the instructions below:

- To **Search Up the Band** for radio stations press and hold ▶▶***forward*** until the search begins *(see note 4 below)*.
- To **Search Down the Band** for radio stations press and hold ◀◀ ***backward*** until the search begins *(see note 4 below)*.

- To **Fine Tune a Station** press ◀◀ ***backward*** briefly to tune down by 50kHz steps. Briefly press ▶▶ **forward** to tune up by 50kHz steps.
- To toggle between Stereo and Mono modes, press Joystick left or right *(see note 2 below)*.
- To **Create a Station Preset** press the ⏎ *Save/Edit* key while the station is playing.
- To **Dial On** without returning to the Preset List, use Joystick up/down to automatically jump to the next or previous station in Preset List.
- To **Return to Station Preset List** press ■ ***stop***.
- To **Mute/Un-mute the Radio** press ▶***play*** while radio is already playing. "Muted" will be displayed in status bar. Press ▶***play*** again to Un-mute the radio.
- To **Exit Radio Mode** press any other direct access key. You can also press ■ ***stop*** while in Preset List selection screen. This will return you to the Home screen.

Notes

(1) The MadPlayer uses the headset cord as the FM antenna. Therefore, you must connect your headset to get the strongest possible signal.
(2) When you are in a fringe area of a particular FM station, the stereo program from that station may have a lot of static. To reduce the noise interference, it may be useful to force the receiver into Mono mode.
(3) Due to the nature of the MadPlayer FM antenna, achieving the best FM radio tuning may require proper environmental conditions—that is, use it outside, keep noisy sources far away from the MadPlayer, unfold the headset cord fully.
(4) The automatic tuner will scan up or down and lock onto the next station. If the automatic tuning does not precisely tune into a radio station, briefly press the backward/forward keys to adjust the tuning.

Editing radio presets

Once the Radio Preset Edit screen is opened:

- To **Select an Entry** use Joystick up/down.
- To **Edit a Selected Entry** use Joystick right to highlight the parameter/attribute to be edited.
- To **Cycle Through Available Choices for Parameters/Attributes After Selecting An Entry,** use Joystick up/down.
- To **Save the Edit Change(s) and Exit** press the ⏎ *Save/Edit* key.
- To **Abort the Edit Change(s) and Exit**, press ■ ***stop*** twice, or press any direct access key.

You also can make changes to as many elements as you want and then save all the changes at once when you exit, or you can exit without saving any of them. The radio preset parameters that can be edited are the name and the frequency.

Renaming a Radio Preset

Use Joystick up/down to select NAME, then use Joystick right to select the current title of the Radio Preset. Once selected, the first letter of the title will begin flashing. Follow the commands below to change the Radio Preset title. The maximum title length is 12 characters.

- To **Erase the Title** press ■ ***stop***. A blinking 'A' will appear on the left. Follow the procedures below to insert/change characters of the title.
- To **Select a Title Character** to be changed, use Joystick left/right. The selected character will blink.
- To **Change a Selected Character** use Joystick up/down to cycle through the letters of the alphabet. When correct character is selected move to next character to be changed, or Save the new title.
- To **Insert an Extra Character** to the left of a selected character, press ▶▶ ***forward.***
- To **Add a Character** at the end of the title, select the last character and press Joystick left.
- To **Delete a Character** first select it, then press ◀◀ ***backward.***
- To **Save the New Radio Preset Title** press the ⮊ ***Save/Edit*** key. This will save the new title and return you to the Radio Presets Directory.
- To **Quit Without Saving Radio Preset Title Changes** press ■ ***stop*** three times. This returns you to the Radio Presets Directory. Or press any direct access key to leave the Radio Preset Edit screen without saving changes.

Changing a Radio Preset Frequency

Use Joystick up/down to select FREQUENCY, then use Joystick right to select the current frequency of the Radio Preset. Once selected, the first digit of the frequency will begin flashing. Follow the commands below to change the Radio Preset Frequency.

- To **Select a Digit** to be changed, use Joystick left/right. The selected digit will blink.
- To **Change a Selected Digit** use Joystick up/down to cycle through the digits. When correct digit is selected move to next one to be changed, or Save the new title.
- To **Save the New Radio Preset Frequency end exit** press the ⮊ ***Save/Edit*** key. This will save the new title and return you to the Radio Presets Directory.
- To **Quit Without Saving Radio Preset Frequency Changes** press ■ ***stop*** two times. This returns you to the Radio Presets Directory. Or press any direct access key to leave the Sample Set Edit screen without saving changes.

KARAOKE MODE

The MadPlayer is a full-function hand-held Karaoke Player that lets you sing to your favorite hits *anywhere*. If you have the MadPlayer, you're ready to party.

[The current version of the MadPlayer supports type 0 Karaoke files (.KAR). Automatic conversion from MIDI Karaoke files of type 1 (Soft Karaoke) is supplied by MadWare™ software. For more information on the conversions that MadWare software provides, please read the MadWare User's Guide.]

Karaoke Songs are downloaded and stored on the MadPlayer's SmartMedia™ Card in exactly the same way as songs in any other supported audio format—MP3, WMA, etc.

And they're played the same way, too! Except that when you play a Karaoke Song on the MadPlayer, the song lyrics are scrolled up to four lines at a time at the bottom of the LCD display screen.

Sing along using the microphone that comes with MadPlayer's headset, or you can plug in a hand-held mic to the MadPlayer jack, and do your thing. Either way, MadPlayer lets you adjust the music Pitch and Tempo to find that perfect fit for your voice. Then it supplies a number of options to let you make your voice whatever you want it to be.

- To **Select and Play a Karaoke Song** refer to Saving/Playing Non-MadPlayer Songs, Page 19. Karaoke Songs are saved in the same SmartMedia™ Card Song Directory as all other songs, and are played the same way.
- To **Play a Karaoke PlayList** refer to Playing Playlists, Page 22. Playlists of Karaoke Songs are controlled and played in exactly the same way as Playlists that contain other types of songs.
- To **Create or Modify Karaoke Playlists** refer to the relevant sections of SONG STORAGE & PLAYBACK, Page 17; and PLAYLISTS, Page 18.
- To **Change Karaoke Song Tempo** press √ *Pitch/Tempo* key together with Joystick right—to speed up, or Joystick left—to slow down.
- To **Change Karaoke Song Pitch** press √ *Pitch/Tempo* key together with Joystick up—to raise pitch, or Joystick down—to lower pitch.
- To **Control Microphone During Karaoke** refer to Microphone Controls, Page 28.

SYSTEM MENU

Using The System Menu

The system Menu allows access to, or modification of, a number of special functions. These functions includes:

- **File Management** of all files stored on the SmartMedia™Card, allowing you to rename, delete, copy, or list song files. It also allows you to modify attributes of certain system files.
- **MadPlayer Configuration**:
  Auto-play (demo mode)
  Power off delay
  Keypad auto-repeat
  Display contrast
  LCD off delay
  Light off delay
  Auto Sample on/off
  Equalizer presets
  Microphone Settings
  Language
  Etc.
- **User Configuration**
  User name
  Custom equalizer presets
  Custom sample effects
- **Upgrades**
  Firmware (Madplayer Software) upgrade
  Sound bank upgrade
- **SmartMedia™Card Formatting**

Each of the above functions represents a sub-menu that is reached through the System Menu. To open the System Menu and choose a sub-menu follow these steps:

- To **Open System Menu** press the ☼ ***System*** direct access key.
- To **Select a Sub-Menu** use Joystick up/down to scroll through the list of Sub-Menus until your choice is in bottom selection box.
- To **Open Selected Sub-Menu** press ► ***play.*** This will open the selection screen for the chosen Sub-Menu. All of the Sub-Menus are explained below.
- To **Exit the System Menu** press ■ ***stop***, this will return you to the Home screen; or press any other Direct Access key to switch directly to another Mode.

File Management

- To **Open the Files Menu** select FILES from the System Menu and press ► ***play***.

Figure 18 shows the opening screen of the Files Menu. The number of files present on the SmartMedia™Card appears on the left side of the status line. The right side indicates the amount of used memory. Below the status line is a separate line listing the amount of free memory left on the card, followed by a list of all the files that are stored on the card.

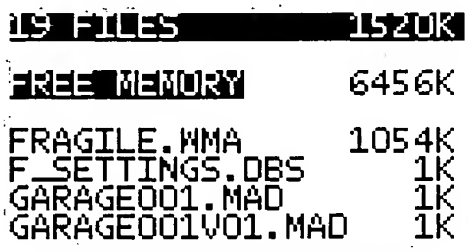


Figure 18: Files Menu Opening Screen

- To **Browse File List** use Joystick up/down.
- To **Scroll to Next Page** press ▶▶ *forward.*
- To **Scroll to Previous Page** press ◀◀ *backward.*
- To **Jump to Top of List** press Joystick left.
- To **Select a File** stop when it is in Reverse Highlight.
- To **Enter File Management** menu for the selected file, press ▶ *play.*

Once a file is opened, the File Management Menu offers a list of actions to perform on the selected file.

- Delete,
- Rename,
- Copy to same card or to another SmartMedia card,
- Change attributes.

The Delete, Rename and Copy functions have already been explained elsewhere in the manual, and will not be covered again here (see Managing Music Files, page 36). To use the "Change Attributes" function follow these steps.

Changing Attributes Function

- To **Modify the File's Attributes** select ATTRIBUTES from the list of File Management actions, and press ▶ *play.* The screen now displays the Attributes edit screen. The name of the file appears in the status line, along with one line per attribute.

Currently, the MadPlayer is only programmed to modify three attributes of files that are stored on a SmartMedia™Card. These attributes are the date, the hidden state (as in a PC) and the "Read Only" designation. By default, all system files (files with a .DBS extension) are "Read Only," and all other files are not.

- To Change the Value of an Attribute, use Joystick up/down to select the attribute parameter—such as READ-ONLY. Then use Joystick right to select the value—such as Yes/No. Use Joystick up/down to change the value.
- To **Save Changes** press the ⮊ *Save/Edit* key. This will save the attribute change and return you to the System Menu.

- To **Abort Attribute Change** press ■ ***stop***. This will discard the change without saving it and return you to the System Menu.

System Configuration

Once CONFIG is selected in the System Menu, the Configuration Menu edit screen is displayed. See Figure 19. Configuration appears in the status line, and below it is the list of configurable parameters.

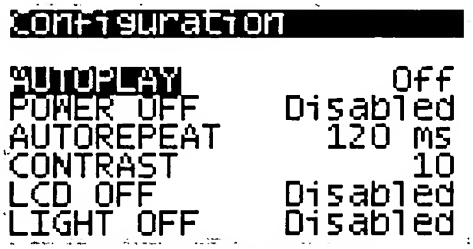


Figure 19: Configuration Menu

Use these commands to navigate and modify the list of parameters :

- To **Browse the List** use Joystick up/down.
- To **Jump to Top Item of List** use Joystick left.
- To **Edit a Parameter**, scroll until item is in Reverse Highlight, then press ► ***play***. Once ► ***play*** is pressed the parameter value begins blinking and is ready for editing.
- To **Change Parameter Value** use Joystick up/down.
- To **Validate Change and Jump** directly to next line of the list, press ► ***play.***
- To **Validate Change and Return** to left side of the edit screen, use Joystick left.
- To **Save Change and Exit** press the ⮊ *Save/Edit* key.
- To **Discard Change** and return to left side of the edit screen, press ■ ***stop.***
- To **Discard Change and Exit** press ■ ***stop*** again.

This table describes the parameters and their configurable values.
Default values are in **bold** type.

Parameter	Values	Description
AUTOPLAY	ON/**OFF**	If AUTOPLAY is ON, the MadPlayer automatically starts playing the first PlayList contained on a SmartMedia card when inserted.
POWER OFF	**Disabled**, 1mn to 60mn in steps of 1mn.	Auto power off delay. The MadPlayer will power off automatically after this delay if no user action is detected.
AUTOREPEAT	40ms to 600ms in steps of 20ms, **120ms**	Keyboard auto-repeat delay in milliseconds. Delay before repeating the corresponding action when a key is pressed continuously.
CONTRAST	0 to 20, **10**	0 = no contrast, 20 = max contrast.
LCD OFF	**Disabled**, 1mn to 60mn in steps of 1mn.	Auto LCD off delay. The MadPlayer will switch the LCD off automatically after this delay if no user action is detected. Pressing any key will switch the LCD back on when off.
LIGHT OFF	**Disabled**, 1mn to 60mn in steps of 1mn.	Auto LCD backlight off delay. The MadPlayer will switch the LCD backlight off automatically after this delay if no user action is detected.
AUTOSAMPLE	**ON**/OFF	If AUTOSAMPLE is ON, the MadPlayer will automatically insert samples while playing eDJ songs. When off, only manual samples will be played (using sample lane/tunnel).
EQ PRESET	**Standard** Woof Hitech Flat User	Presets for 4-band equalizer. Standard, Woof, HiTech and Flat are factory presets and fixed. User preset can be configured by the User via the User Config menu.
MIC STATE	ON/**OFF**	Microphone input is ON or OFF.
MIC VOLUME	0 to 31, **5**	Microphone volume.
ECHO LEVEL	0 to 127, **39**	Level of echo applied to microphone input
ECHO TIME	0 to 127, **63**	Microphone echo delay. 0 shortest, 127 longest.
ECHO FEEDBK	0 to 127, 79	Echo feedback: 0 minimum feedback, 127 maximum feedback.
REC FORMAT	PCM 11K PCM 16K **PCM 22K**	Format used to store recorded samples: PCM: PCM, 16 bits mono HQFADPCM: High Quality Fast ADPCM

	ADPCM 11K ADPCM 16K ADPCM 22K.	11,16,22 kHz : sampling rate. (Note that other formats are supported in play mode but cannot be recorded on the MadPlayer).
SORT FILES	**BY NAME** BY TYPE	Criterion used to sort files when displaying a list: by name (alphabetically) or by type (songs, samples, lists...).
SORT PRESETS	**BY NAME** BY FREQ	Criterion used to sort radio presets: by name (alphabetically) or by frequency.
LANGUAGE	**ENGLISH** FRANCAIS ESPANOL	Language used for the menus.
MIDI DSP FW	String	READ ONLY. DSP Firmware version
SOUND BANK	String	READ ONLY. Sound bank file name
SERIAL NUM	10 Digit Number	READ ONLY. Unique serial number of the MadPlayer
HW VERSION	String	READ ONLY. Hardware revision number
SW VERSION	String	READ ONLY. Firmware revision number

Firmware Upgrade

Firmware is the embedded software that MadPlayer uses to perform many of its functions. Periodically, new versions of this firmware will be posted for downloading from the www.MadPlayer.com website. The website will also provide specific instructions on how to conduct the download to your SmartMedia Card.

In order to initiate a firmware upgrade, two files are needed :

- One file is a small program file which allows you to execute the firmware upgrade function ("Loader program file").
- File name extension is DBL (eg ."DB1UPG_V3.DBL").
- The second file contains the new firmware code to be programmed into MadPlayer's memory ("Firmware file").
- File name extension is DBF (eg ."P1FW.DBF").

You must already have stored (for example, after downloading from Madware's web site to the SmartMedia Card) at least one Loader program file and one Firmware file on your current SmartMedia Card before you can open the Firmware Upgrade Menu. Otherwise, a warning message will be displayed and the MadPlayer will return to the System Menu.

After choosing UPGRADES from the System Menu, select UPGRADE FW and the Upgrade Firmware menu will appear, listing the firmware upgrade file(s) that you have downloaded to the SmartMedia™ Card.

- To **Select a Firmware Upgrade File**, use Joystick up/down until the correct file is in Reverse Highlight.
- To **Begin the Firmware Upgrade**, press ▶***play.***

A confirmation screen is now displayed.

- To **Confirm Your Upgrade Choice** use Joystick up/down to select YES or NO.
- To **Continue the Upgrade**, after you have selected YES for confirmation, press ▶***play.***
- To **Abort Upgrade and Return** to the Upgrade Firmware menu press ■ ***stop.***
- To **Abort Upgrade and Exit** to the Upgrades Menu press ■ ***stop*** again.

After a Firmware Upgrade is completed, simply switch the MadPlayer off, and then on again: the MadPlayer now runs the new version of firmware.

Format SmartMedia™Card

After choosing FORMAT CARD from the System Menu, the Format SmartMedia™Card menu will appear. This initial screen proposes a volume name for the card that is to be formatted.

You can simply accept the proposed volume name, and skip to the second step below—**Begin Formatting**, or you can choose to create a new volume name, first.

- To **Create a New Volume Name** press ■ ***stop,*** to return to the System Menu. Then follow the instructions for Renaming Song Files, page 22. The exact same procedure is used to rename a Volume Name for a SmartMedia™ Card. When the renaming is completed return to this menu and continue to the next step, below.
- To **Begin Formatting** press ▶ ***play*** when the Volume Name is in Reverse Highlight. A confirmation screen is now displayed.
- To **Confirm Your Formatting Choice** use Joystick up/down to select YES or NO.
- To **Continue Formatting**, after you have selected YES for confirmation, press ▶***play.*** The SmartMedia™Card will now be formatted.
- To **Quit the Formatting Process,** after you have selected NO to confirmation, press ▶***play.*** This will return you to the System Menu.
- To **Quit the Fomatting Process Immediately**, press ■ ***stop***. This will return you to the System Menu.

When formatting is completed, the LCD screen reads "CARD FORMATTED," then creates the system files (some messages appear on the status line). After a few seconds, you're back in business with a brand new card.

User Configuration

- To **Open the User Configuration Menu** select USER CONFIG from the System Menu and press ▶ ***play***

Defining a User Name

To **Define a User name**, select USER NAME from the System Menu and press ▶ ***play***. This will store your log name on the SmartMedia card. The log name will be used in some screens (for instance in the Standard Home Screen welcome message).

Setting the Digital Equalizer parameters

MadPlayer comes with 4 equalizer (EQ) factory presets:

- Flat: no EQ effect.
- Standard: slight boost of lower and higher frequencies.
- Woof: bass frequencies boost.
- Hitech: used mostly in Techno styles.

In addition, a User preset allows you to define your own values for the equalizer's 4 bands of frequencies.

- Use the joystick up/down:left/right to edit EQ values of each parameter. While playing a song, you can alternatively use the User Preset shortcut (holding down √ *Pitch/Tempo*, press ☼ *System* key).
- Press the ⇨ *Save/Edit* key to save the user settings.

You can select the active EQ preset either by using the System Configuration menu (see page 46), or by using the EQ preset shortcut (holding down ≡ *FX* , press ☼ *System* key).

Defining Custom Effects

MadPlayer allows you to apply sound effects to the samples. There are 6 factory-defined effect presets:

- No effect.
- Lo-voice: plays the song with slower speed and lower pitch.
- Reverb: high reverb.
- Hi-voice: plays the song with faster speed and higher pitch.
- Doppler: varies the sound from hi-voice to lo-voice.
- Wobbler: simulates several voices with effects.

In addition, Custom Effect presets allow you to define your own values for the FX generator parameters. To create a new Custom Effect :

- Select the SYSTEM menu. Once you're in the SYSTEM menu, select USER CONFIG, and from that menu, choose CUST EFFECTS.
- Press ▶▶***forward*** to create a new template.

- Select the default parameters by choosing the pre-defined ("Pre-load") effect from which you want to derive your new effect parameters. For instance, if you want to modify the Lo-voice effect and create a new effect based on Lo-voice effect parameters, use joystick up/down and press ▶ ***play*** when "Lo-voice" appears at the bottom of the screen.
- You can then use the edit screen to:
  - define the effect name
  - select the sample on which you want to test the effect
  - select which sub-effect functions you want to use (doppler on/off, wobbler on/off, flange on/off, echo on/off, reverb on/off)
  - set the parameters for the selected functions (on) using joystick up/down/left/right
- Test the effect by pressing the the *FX* key. MadPlayer will play the selected sample (see above) with current effect parameters.
- Once you like the effect settings, save them by pressing the *Save/Edit* key.

TECHNICAL APPENDIX

Sound Bank Upgrade Procedure

The sound bank contains the digital sound samples used by the MadPlayer synthesizer to generate the music MadPlayer plays. Periodically, new versions of the sound bank will be posted for downloading from the www.MadPlayer.com website. The website also will provide specific instructions on how to download the new versions to your SmartMedia™ Card.

You need two files to start a Sound Bank upgrade:

- File 1: The "Sound Bank Loader" file – a small program file which allows you to execute the sound bank upgrade function
- File name extension is SML (eg ."SBUPG_V3.SML").
- File 2: the "Sound Bank" file, which contains the new firmware code to be programmed in MadPlayer's memory.
- File name extension is SMF (eg ."SB1.SMF").

You must have already stored (for example, after downloading from Madware's web site to the SmartMedia™Card) at least one Sound Bank Loader file and one Sound Bank file on your current SmartMedia Card before you can open the Sound Bank. Upgrade Menu. Otherwise, a warning message will be displayed and the MadPlayer will return to the System Menu.

After you have the Sound Bank Loader file and Sound Bank file on your SmartMedia card, here's how you perform the upgrade:

- Choose UPGRADES from the System Menu. Once you're in the UPGRADES Menu, select UPG SND BANK. The Upgrade Sound Bank menu will then appear, listing the sound bank file(s) that you have downloaded to the SmartMedia™ Card.
- To **Select a Sound Bank Upgrade File**, use Joystick up/down until the correct file is in Reverse Highlight.
- To **Begin the Sound Bank Upgrade**, press ▶***play.*** A confirmation screen is now displayed.
- To **Confirm Your Upgrade Choice** use Joystick up/down to select YES or NO.
- To **Continue the Upgrade**, after you have selected YES for confirmation, press ▶***play.***
- To **Abort Upgrade and Return** to the Upgrade Sound Bank menu press ■ ***stop.***
- To **Abort Upgrade and Exit** to the Upgrades menu press ■ ***stop*** again.

After a Sound Bank Upgrade is completed, a completion message is displayed. Simply press any key to return to System Menu. The MadPlayer will now use the new version of sound bank.

USB/PC Interface

The MadPlayer features a standard USB interface. This interface offers a high-speed connection to PC computers that allows you to transfer files (songs, samples, etc...) to and from the SmartMedia card in the MadPlayer or to perform a firmware upgrade.

WARNING: The MadPlayer connects to the PC USB port only with the supplied cable. Attempting to use another cable for this purpose may damage the MadPlayer and/or the PC.

To communicate with the MadPlayer and manage the content of the SmartMedia card, it is necessary to use MadWaves' MadWare, supplied with the MadPlayer. For more details on the functions available when MadPlayer is connected to a PC running MadWare, please refer to the MadWare User's Guide, included on the MadWare CD supplied with the MadPlayer.

Before MadWare can be used to communicate with the MadPlayer it is necessary to install the USB driver. Please refer to the MadWare Installation instructions to install the USB driver.

Connecting the MadPlayer to your PC

- Power up the MadPlayer and go to Home mode. **Connecting the MadPlayer to your PC when MadPlayer is not in the Home mode may result in incorrect operation of the USB link.**
- Connect the USB cable to one of the USB ports of your PC.
- Connect the USB cable to the MadPlayer. [Note: The 15-pin connector is keyed so that you can only connect it in one position—with the metal shield facing downward.

You'll know that you've installed the driver successfully when MadPlayer displays a flashing PC icon in the home screen. This flashing PC icon is the sign that the MadPlayer has been configured on the USB bus and that the driver is loaded.

Figure 20: Home screen with USB active

If the flashing PC icon does not appear when connecting the MadPlayer to the PC, disconnect the USB cable, power up the MadPlayer, go to Home mode and reconnect the USB cable.
The flashing PC icon must always be displayed in order to be able to use the MadWare software application.

Using MadWare

Important note: In order to communicate with the MadWare, the MadPlayer must be in the Home mode. It is important to always come back to Home mode before connecting the USB cable to the MadPlayer and your PC to work with the MadWare; and, you must stop working with the MadWare, and disconnect the USB before exiting MadPlayer's Home mode. Also you cannot insert or remove a SmartMedia card when working with the MadWare. Failure to follow these instructions may damage the SmartMedia card or produce unpredictable behavior in the software and hardware.

We can't stress this enough: Make sure the MadPlayer is connected to your PC, in the Home mode and properly configured on the USB bus (flashing PC icon) before starting the MadWare.

Remote link session with the MadPlayer

Once you've connected the MadPlayer to your PC per the instructions above, you can begin working with the MadWare. For instructions on Connecting and Disconnecting a session with the MadWare, please refer to the MadWare User's Guide.
When a session begins, the MadPlayer displays the "Remote" screen. An image of the PC and the MadPlayer is shown with a dot going to and from the PC, to indicate an active connection.

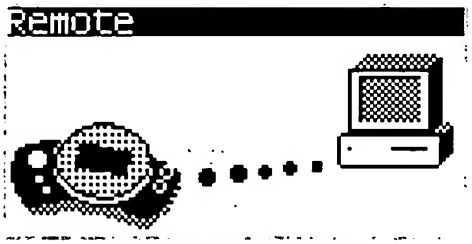


Figure 21: Remote screen

Disconnecting from the MadWare:
When you're done working with the MadWare, follow the directions in the MadWare User's Guide to disconnect. When you disconnect, the MadPlayer comes back to the Home screen with the flashing PC icon. In this state, it is safe to use other functions of the MadPlayer or to disconnect the USB cable.

Music Pattern Numbering

Music patterns are the Generative element of songs created by the MadPlayer. For example, if you are in Tunnel Mode for the Drums lane, each time you press Joystick right, MadPlayer will compose a new drum pattern and save the previous one. The same type of pattern creation occurs when you are in Tunnel Mode for Bass, Riff or Lead. Every pattern the MadPlayer creates is completely new, and it will keep creating new patterns for as long as you request them. That's great, right? But it can also be hard to keep track of them all. That's why MadPlayer has Music Pattern Numbering!

Within the Tunnel Mode for each instrument lane, every music pattern that is composed by MadPlayer is assigned a number, from 1 to 65535. The number is displayed in the status line while the pattern is playing, to help you connect the number to what you're hearing, so that you can go back to find that pattern after having tried some/a lot of others.

All pattern numbers and patterns, are only saved during the current interactive session—that means during the song that's currently playing. Once you save the song, or discard it, the numbered patterns in all the Instrument Lanes are also discarded. For that reason, when you go back to play a song that has been saved, it will contain the patterns it had when you saved it, but if you start revising the song again by generating new drum patterns, for example, the pattern in the saved song would now be pattern 1, no matter what number it was assigned in the previous interactive session.

Although the MadPlayer can keep assigning numbers to newly created patterns almost indefinitely (max = 65535), it can only "remember" up to 16 previously composed patterns (for each instrument lane) during the current interactive session. That means that if the current drum pattern, for example, is 25, then you can listen to patterns from 10 to 25 by using the Joystick to browse through previous patterns. But patterns 1-9 would no longer be available for use.

Keeping in mind this memory limit of 16 patterns, you can still cycle through many more patterns than that if you utilize the Discard Pattern function. As mentioned above, if you listen to a pattern and then use Joystick right, a new pattern is begun and the previous pattern is saved. If, instead, you use Joystick down to begin a new pattern, the previous pattern will not be saved, and a number will not be added to the pattern list. By utilizing this function you can listen to hundreds of patterns and only save the most promising 16 of them.

- To **Save the Current Pattern** and start a new music pattern, use Joystick right.
- To **Discard the Current Pattern** and start a new music pattern, use Joystick down.
- To **Browse the Saved Patterns** use Joystick left/right.

Music Styles and Sub-Styles

You can see all the music styles available for your use by browsing in e-DJ menu. The styles displayed are for the current versions of the firmware and sound bank. Some of the styles are grouped as "Mixes" (eg TECHNO Mix, URBAN Mix, etc.). When you select a Mix from the list, MadPlayer creates (composes) songs by randomly picking one of styles included in that Mix. The list, below, shows the Mixes and their elements (basic styles) for the current Firmware and Sound Bank release. [Note: Styles/mixes may change from those listed below, depending on the firmware release and the sound bank in your MadPlayer. You can download new releases/sound banks from www.MadWaves.com web site.]

[Note: In the e-DJ menu, the Mix names are in CAPITAL letters, while elementary styles are in initial cap, lower-case.]

Mix types also are used when creating Sample Sets to make sure that a Sample Set will only be used for a given Mix of Styles.
For example, if you select "TECHNO" as the specific mix for a given Sample Set, this Sample Set will only be used for e-DJ songs or MadSongs of one of the following styles: House, Garage, Trance, DnBass, Jungle.

Mixes Description

TECHNO MIX
Gathers House, Garage, Trance, DnBass and Jungle styles.

URBAN MIX
Gathers Hiphop, Rap, RnB, Downbeat and Ragga styles.

COOL MIX
Gathers Ballad, Bossa, Newage and Ambient styles.

MANGA
Contains only Manga style.

Styles Description

House
The most famous and easiest techno to listen to. House uses the standard dance rhythm with a moderate tempo (open hats in between the on-the-beat bass drum). Sounds are exclusively electronic but not too weird. Some chord progressions and riffs are still close to Disco.

Garage
Historically Garage is a European refresh of Chicago House. Further away from disco, tempos are faster and synthesizers are more filtered. Static chord changes provide an all-night-long rave atmosphere.

Trance
The classic Techno party style. Lots of trippy sounds and noises combined with large ambient pads (synths) give the typical psychedelic atmosphere. Over 145 tempos provide no drum breaks to let the music (and the dancers) breathe.

DnBass (Drum and Bass)
Drum and Bass is characterized by the eccentric speeded-up funky drum patterns. Running at fast 160 tempos, drum textures are synthesized and filtered, giving many unusual effects. The bass line comes from Reggae, flowing smoothly or pumping energetically. Climate lays on oppositions, where syncopated beats (i.e. off the measure) meet space synths, sometimes interwoven with smooth jazz and unexpected sounds.

Jungle
Like Drum and Bass, Jungle has a non-linear and polyrhythmic approach to the beat, and the digitized Reggae influence is still noticeable on repetitive up-front bass lines. Known as the UK's response to US' Old School HipHop, this style reflects instability and defiance of established rules. The nervous funky rhythm with weird electronic sounds is another kind of dance and listening music, for those fed up with the standard Disco beat.

Manga
Inspired by Japanese comics and animation, this style is an 80's Disco played with electronic instruments. Easy listening harmony, joyful melodies, happy dance grooves. Spontaneously dancing, tempo around 125 is synchronized with the heart of the dancer. Works really well with your favorite samples.

Hiphop
This is where you'll find the famous breakbeat, delirious sound FXs, scratches and sometimes natural instruments and chords. Rappers often perform with Hiphop, sampling and remixing to fit the beats with their poetry. A great style for improvisation and vocal gymnastics.

Rap
We define Rap as a rough Hiphop style. Drums are more powerful, sound FXs and noises harder. Orchestration is as creative as possible, but also provides the repetition rappers prize. Rap also contains the elements of Hiphop remixed in a tight, static way to emphasize the rapper's voice.

RnB
Coming from the late 90's Pop songs (and not to be confused with good ol' Rhythm & Blues by Otis Redding), the third millennium RnB keeps the breakbeat spirit of Hiphop. Nevertheless RnB grooves are a sophisticated mix of Funk, Reggae, and Soul with a lot

of frenetic rhythmical and instrumental influences from Latin to Dance. Harmony is subtle, mostly based on Gospel and Soul chord progressions.

Downbeat

Basically a soft "you-can-sing-to-it" Hiphop style. Warm grooves are softer than in Rap, orchestration is closer to RnB with less sound FX. No weird stabs here but big mellow bass sounds. Chords borrowed from Soul and Jazz create a warm and peaceful atmosphere.

Ragga

In our context Ragga is the epitome of a music style with deep roots in the Reggae scene. A Ragga song may sound like traditional Caribbean, 70's Pop Reggae hit, up-tempo Ska and even experimental electronic Dub. Listen to the laid back, fat bass and enjoy the sun, ya mon!

Ballad

Far away from the dance floor, Ballad is a soft background music style. Most of the instruments are natural, orchestrated in a basic song structure (intro, theme, chorus, ending). Mid tempos provide a few Slow Rock clichés, especially on guitars and basses.

Bossa

Bossa is a background music style based on Latin rhythms. You will also hear a lot of acoustic instruments, sometimes playing more sophisticated lines to fit the exotic percussion grooves.

NewAge

Here again, NewAge is meant to be heard rather than listened to. Often described as relaxation music, this style explores free and ethereal notes played by instruments such as piano, bells, acoustic guitar or pan-flute. Soothing, mellow and all-embracing, NewAge is designed to create a sense of peace for the listener.

Ambient

Ambient music defines a soundscape (a musical landscape) which is peaceful and non-invasive. Mostly synthetic, Ambient features slow electronic or ethnic cyclic rhythms. The lengthy evolution of themes creates a global atmospheric texture, usually expansive and cinematic in feel.

Supported Audio Formats

Windows Media Audio: The current release of the MadPlayer supports only files coded at a sampling rate of 22.05kHz, 32kHz or 44.1kHz with a bit rate not exceeding 128Kbps.

MIDI & KARAOKE: The current version of the MadPlayer supports MIDI and Karaoke files (.KAR) of type 0. Automatic conversion from MIDI files of type 1 and MIDI Karaoke files of type 1 (Soft Karaoke) is supplied by MadWare software. For more information on conversions that MadWare software provides, please read the MadWare User's Guide.

Audio sample: There are two different audio formats for samples

- PCM : Digital audio without compression (Similar to .WAV Audio files).
  - Sampling frequencies supported (record) : 11.025kHz, 16kHz, 22.05 kHz.
  - Sampling frequencies supported (playback) : 4kHz, 8kHz, 11.025kHz, 16kHz, 22.05 kHz, 32kHz, 44.1kHz.
- HQF-ADPCM : MadWaves Proprietary compressed audio samples
  - Sampling frequencies supported (record) : 11.025kHz, 16kHz, 22.05 kHz.
  - Sampling frequencies supported (playback) : 4kHz, 8kHz, 11.025kHz, 16kHz, 22.05 kHz.

The first one (PCM) is higher quality but takes approximately 3.5 times more memory. For example a 16-bit, mono, 5-second sample @ 22kHz will take approximately 215KBytes in WAV format while it will only take 60KBytes in MadWaves proprietary HQF-ADPCM format.

Sound Bank Contents

MadPlayer's soundbanks contain over 400 sounds and instruments! The following tables describe the contents of the BKDBT44 MadPlayer Sound Bank, in the standard MIDI description format. To check if your MadPlayer uses this bank, press ♂*System* direct key, then select CONFIG, and use Joystick up/down to see the loaded SOUND BANK name. Other banks may be available on the www.MadWaves.com web site for download. Check them out !

PC: Program change
C0 : Controller 0 value (zero for General Midi capital sounds)
To select variation send CTRL 0, then PC

BKDBT44 General MIDI instruments

PC	GENERAL MIDI	C0	VARIATION	C0	VARIATION
0	Acoustic Grand Piano	1	Piano String Combi		
1	Bright Acoustic Piano				
2	Electric Grand Piano				
3	Honky-tonk Piano				
4	Electric Piano 1	1	Jazz Vintage EP1	2	Jazz Vintage EP2
		3	Jazz Vintage Vibes 1	4	Jazz Vintage Vibes 2
5	Electric Piano 2	1	Rock Vintage EP 1	2	Rock Vintage EP 2
		3	Rock Vintage Vibes 1	4	Rock Vintage Vibes 2
		5	Rock Vintage Vibes 3		
6	Harpsichord	1	RnB Harpsichord		
7	Clavi				
8	Celesta				
9	Glockenspiel				
10	Music Box				
11	Vibraphone				
12	Marimba				
13	Xylophone				
14	Tubular Bells				
15	Dulcimer				
16	Drawbar Organ	1	Soft Solo Organ		
17	Percussive Organ	1	Jazz Organ		
18	Rock Organ	1	Vocoder Organ		
19	Church Organ				
20	Reed Organ				
21	Accordion				
22	Harmonica				
23	Tango Accordion				
24	Acoustic Guitar (Nylon)				
25	Acoustic Guitar (Steel)				
26	Electric Guitar (Jazz)	1	Django Jazz	2	Pat Jazz
27	Electric Guitar (Clean)	1	HQ Clean Guitar	2	Fat Clean
		3	Fun Funk	4	Ragga Chord 1

		5	Ragga Chord 2		
28	Electric Guitar (Muted)	1	Pat Muted		
29	Overdriven Guitar				
30	Distortion Guitar				
31	Guitar Harmonics				
32	Acoustic Bass	1	Acoustic Bass 2	2	Acoustic Bass 3
33	Finger Bass	1	Finger Bass 2	2	FM Bass
34	Picked Bass	1	Picked Bass 2		
35	Fretless Bass				
36	Slap Bass 1				
37	Slap Bass 2	1	Slap Bass 2		
38	Synth Bass 1				
39	Synth Bass 2				
40	Violon				
41	Viola				
42	Cello				
43	Contrabass				
44	Tremolo String				
45	Pizzicato Strings				
46	Orchestral Harp				
47	Timpani				
48	String Ensemble 1				
49	String Ensemble 2				
50	Synth Strings 1				
51	Synth Strings 2				
52	Choir Aahs				
53	Voice Oohs				
54	Synth Voice	1	HQ Synth Voice		
55	Orchestra Hit				
56	Trumpet	1	Trumpet 2		
57	Trombone				
58	Tuba				
59	Muted Trumpet	2	Muted Trpt 2		
60	French Horn				
61	Brass Section	1	Brass Fall		
62	Synth Brass 1				
63	Synth Brass 2				
64	Soprano Sax	1	Soprano with breath		
65	Alto Sax	1	Alto with breath		
66	Tenor Sax	1	Tenor with breath		
67	Barytone Sax	1	Barytone with breath		
68	Oboe				
69	English Horn				
70	Basson				
71	Clarinet				
72	Piccolo				
73	Flute	1	Breathless Flute		
74	Recorder				
75	Pan Flute				
76	Bottle Blow				
77	Shakuhachi				
78	Whistle				
79	Ocarina				

80	Lead 1 (Square)				
81	Lead 2 (Sawtooth)				
82	Lead 3 (Calliope)				
83	Lead 4 (Chiff)				
84	Lead 5 (Charang)				
85	Lead 6 (Voice)				
86	Lead 7 (Fifhts)				
87	Lead 8 (Bass + Lead)	**1**	Monophonic 1 (dirty)	**2**	Monophonic 2 (sweet)
		3	Monophonic 3 (clean)		
88	Pad 1 (New Age)				
89	Pad 2 (Warm)				
90	Pad 3 (Polysynth)				
91	Pad 4 (Choir)				
92	Pad 5 (Bowed)				
93	Pad 6 (Metallic)				
94	Pad 7 (Halo)				
95	Pad 8 (Sweep)				
96	FX 1 (Rain)				
97	FX 2 (Soundtrack)				
98	FX 3 (Crystal)				
99	FX 4 (Atmosphere)				
100	FX 5 (Brightness)				
101	FX 6 (Goblins)				
102	FX 7 (Echoes)				
103	FX 8 (Sci-fi)				
104	Sitar				
105	Banjo				
106	Shamisen				
107	Koto				
108	Kalimba				
109	Bag Pipe				
110	Riddle				
111	Shanai				
112	Tinkle Ball				
113	Agogo				
114	Steel Drums				
115	Woodblock				
116	Taiko Drum				
117	Melodic Tom				
118	Synth Drum				
119	Reverse Cymbal				
120					
121					
122	Seashore				
123					
124	Chimes				
125					
126	Applause				
127					

BKDBT44 Drum Sets

PC	Drum Kit	Content
0	Standard Set	Default acoustic drum instruments with Bkdbt44 Acoustic Extension
16	Power Set	Gated snare 1
24	Electronic Set	Default electronic drum instruments with Bkdbt44 Electro Extension
32	Jazz Set	Jazz snares
40	Brush Set	Brush snares ('tap' and 'swirl') and hand clap ('slap')
103	FXs and Percussions Set	Bkdbt44 FXs and Percussions Extension Set

BKDBT44 General MIDI Drum Instruments

	Program 0 : STANDARD SET	Program 16 : POWER SET	Program 24 : ELECTRONIC SET	Program 32 : JAZZ SET	Program 40 : BRUSH SET
35 – B2	Kick drum2	Kick drum2	**Electronic Bass Drum 1**	Kick drum2	Kick drum2
36 – C3	Kick drum1	Kick drum1	**Electronic Bass Drum 2**	Kick drum1	Kick drum1
37 – C#3	Side Stick	Side Stick	Side Stick	Side Stick	Side Stick
38 – D3	Snare Drum 1	**Gated Snare**	**Electronic Snare Drum 1**	**Jazz Snare Drum 1**	**Brush Tap**
39 – D#3	Hand Clap	Hand Clap	Hand Clap	Hand Clap	**Brush Slap**
40 – E3	Snare Drum 2	Snare Drum 2	**Electronic Snare Drum 2**	**Jazz Snare Drum 2**	**Brush Swirl**
41 – F3	Low Floor Tom	Low Floor Tom	**Electronic Low Tom 2**	Low Floor Tom	Low Floor Tom
42 – F#3	Closed Hi Hat	Closed Hi Hat	**Electronic Closed Hi-Hat**	Closed Hi Hat	Closed Hi Hat
43 – G3	High Floor Tom	High Floor Tom	**Electronic Low Tom 1**	High Floor Tom	High Floor Tom
44 – G#3	Pedal Hi-Hat	Pedal Hi-Hat	**Electronic Pedal Hi-Hat**	Pedal Hi-Hat	Pedal Hi-Hat
45 – A3	Low Tom	Low Tom	**Electronic Mid Tom 2**	Low Tom	Low Tom
46 – A#3	Open Hi-Hat	Open Hi-Hat	**Electronic Open Hi-Hat**	Open Hi-Hat	Open Hi-Hat
47 – B3	Low-Mid Tom	Low-Mid Tom	**Electronic Mid Tom 1**	Low-Mid Tom	Low-Mid Tom
48 – C4	Hi Mid Tom	Hi Mid Tom	**Electronic High Tom 2**	Hi Mid Tom	Hi Mid Tom
49 – C#4	Crash Cymbal 1	Crash Cymbal 1	Crash Cymbal 1	Crash Cymbal 1	Crash Cymbal 1
50 – D4	High Tom	High Tom	**Electronic High Tom 1**	High Tom	High Tom
51 – D#4	Ride Cymbal 1	Ride Cymbal 1	Ride Cymbal 1	Ride Cymbal 1	Ride Cymbal 1
52 – E4	Chinese Cymbal	Chinese Cymbal	Chinese Cymbal	Chinese Cymbal	Chinese Cymbal
53 – F4	Ride Bell	Ride Bell	Ride Bell	Ride Bell	Ride Bell
54 – F#4	Tambourine	Tambourine	**Electronic Tambourine**	Tambourine	Tambourine
55 – G4	Splash Cymbal	Splash Cymbal	Splash Cymbal	Splash Cymbal	Splash Cymbal
56 – G#4	Cowbell	Cowbell	Cowbell	Cowbell	Cowbell
57 – A4	Crash Cymbal 2	Crash Cymbal 2	Crash Cymbal 2	Crash Cymbal 2	Crash Cymbal 2
58 – A#4	Vibraslap	Vibraslap	Vibraslap	Vibraslap	Vibraslap
59 – B4	Ride Cymbal 2	Ride Cymbal 2	Ride Cymbal 2	Ride Cymbal 2	Ride Cymbal 2
60 – C5	Hi Bongo	Hi Bongo	Hi Bongo	Hi Bongo	Hi Bongo
61 – C#5	Low Bongo	Low Bongo	Low Bongo	Low Bongo	Low Bongo
62 – D5	Mute Hi Conga	Mute Hi Conga	**Electronic High Conga**	Mute Hi Conga	Mute Hi Conga
63 – D#5	Open Hi Conga	Open Hi Conga	**Electronic Mid Conga**	Open Hi Conga	Open Hi Conga
64 – E5	Low Conga	Low Conga	**Electronic Low Conga**	Low Conga	Low Conga
65 – F5	High Timbale	High Timbale	High Timbale	High Timbale	High Timbale
66 – F#5	Low Timbale	Low Timbale	Low Timbale	Low Timbale	Low Timbale
67 – G5	High Agogo	High Agogo	High Agogo	High Agogo	High Agogo
68 – G#5	Low Agogo	Low Agogo	Low Agogo	Low Agogo	Low Agogo
69 – A5	Cabasa	Cabasa	Cabasa	Cabasa	Cabasa
70 – A#5	Maracas	Maracas	Maracas	Maracas	Maracas
71 – B5	Short Whistle	Short Whistle	Short Whistle	Short Whistle	Short Whistle
72 – C6	Long Whistle	Long Whistle	Long Whistle	Long Whistle	Long Whistle
73 – C#6	Short Guiro	Short Guiro	Short Guiro	Short Guiro	Short Guiro

74 – D6	Long Guiro	Long Guiro	Long Guiro	Long Guiro	Long Guiro
75 – D#6	Claves	Claves	Claves	Claves	Claves
76 – E6	Hi Wood Block	Hi Wood Block	Hi Wood Block	Hi Wood Block	Hi Wood Block
77 – F6	Low Wood Block	Low Wood Block	Low Wood Block	Low Wood Block	Low Wood Block
78 – F#6	Mute Cuica	Mute Cuica	Mute Cuica	Mute Cuica	Mute Cuica
79 – G6	Open Cuica	Open Cuica	Open Cuica	Open Cuica	Open Cuica
80 – G#6	Mute Triangle	Mute Triangle	Mute Triangle	Mute Triangle	Mute Triangle
81 – A6	Open Triangle	Open Triangle	Open Triangle	Open Triangle	Open Triangle
82 – A#6	Shaker	Shaker	Shaker	Shaker	Shaker
83 – B6	Jingle Bell	Jingle Bell	Jingle Bell	Jingle Bell	Jingle Bell
84 – C7	Chimes	Chimes	Chimes	Chimes	Chimes

BKDBT44 Drum set extension instruments

	Program 0 : MAD ACOUSTIC	Program 24 : MAD ELECTRO
Kicks		
1 – C#0	Da kick	Sub bass drum
2 – D0	Shot bass drum	Ice bass drum
3 – D#0	Pump bass drum	Big kick
4 – E0	Low frequency bass drum	RnB kick
5 – F0	Hard kick	Snoop bass drum
6 – F#0	Coast bass drum	Running bass drum
7 – G0	Drum machine bass drum 1	West coast bass drim
8 – G#0	Drum machine bass drum 2	Digital bass drum
9 – A0	Ragga bass drum 1	Musical bass drum
10 – A#0	Ragga bass drum 2	Gated kick 1
11 – B0	Hiphop bass drum 1	Gated kick 1
12 – C1	Hiphop bass drum 2	Mono bass drum
13 – C#1	Standard kick drum 1	Lost bass drum
14 – D1		Electro bass drum
15 – D#1		Tight bass drum
16 – E1		Tough kick
17 – F1		Synth bass drum
18 – F#1		Techno bass drum 1
19 – G1		Techno bass drum 2
20 – G#1		Techno bass drum 3
21 – A1		Jungle bass drum 1
22 - A#1		Jungle bass drum 2
Rim Shots		
24 – C2	Standard side stick	Jungle rim shot
25 - C#2	Jim's rim shot	Hard rim shot
26 – D2	RnB rim shot	
Claps		
30 – F#2	Standard clap	Panoramic clap
31 – G2	Straight clap	Dual clap
32 – G#2	Hard clap	Electro clap
33 – A2		Gun clap
34 – A#2		
Percussions		
85 - C#7	Real shaker 1	
86 – D7	Real shaker 2	
87 - D#7	Real shaker 3	
88 – E7	Real shaker 4	
89 – F7	Real shaker 5	
90 - F#7	Real shaker 6	
91 – G7	Real shaker 7	
92 - G#7	Real shaker 8	
93 – A7	Piccolo snare 1	
94 - A#7	Piccolo snare 2 (ghost note)	

95 – B7	Piccolo snare 3 (roll)	
Snares		
96 – C8	Swing snare	Hard snare 1
97 - C#8	Bright snare 1	Hard snare 2
98 – D8	Bright snare 2	Electric snare
99 - D#8	Hiphop snare	Prince snare
100 – E8	Standard snare 1	Straight snare
101 – F8	Standard snare 2	Joker snare
102 - F#8	Jazz snare 1	Virtual snare
103 – G8	Jazz snare 2	New-York snare
104 - G#8	Brush tap	Beach snare
105 – A8	Brush swirl	Rim shot snare
106 - A#8	Jungle snare 1	North snare
107 – B8	Jungle snare 2	Drum machine snare 1
108 – C9	Ragga snare 1	Drum machine snare 2
109 - C#9	Ragga snare 2	Drum machine snare 3
110 – D9	Timbale snare	Techno snare
111 - D#9		Reverb electro snare
Hi-Hats		
116 - G#9	Standard closed Hi Hat	Hiphop closed Hi Hat
117 – A9	Standard pedal Hi-Hat	Hiphop pedal Hi-Hat
118 - A#9	Standard open Hi-Hat	Hiphop open Hi-Hat
119 – B9	New-York closed Hi Hat	Blues closed Hi Hat
120 – C10	New-York pedal Hi-Hat	Rap pedal Hi-Hat
121 - C#10	New-York open Hi-Hat	Standard open Hi-Hat
122 – D10	Ragga closed Hi Hat	Drum machine closed Hi Hat
123 - D#10	Ragga pedal Hi-Hat	Drum machine pedal Hi-Hat
124 – E10	Ragga open Hi-Hat	Drum machine open Hi-Hat

	Program 103 : **MAD FX & PERCUSSIONS**
Percussions	
1 – C#0	Real shaker 1
2 – D0	Real shaker 2
3 – D#0	Real shaker 3
4 – E0	Real shaker 4
5 – F0	Real shaker 5
6 – F#0	Real shaker 6
7 – G0	Real shaker 7
8 – G#0	Real shaker 8
9 – A0	Big shaker 1
10 – A#0	Big shaker 2
11 – B0	Big shaker 3
12 – C1	Big shaker 4
13 – C#1	Big shaker 5
14 – D1	Big shaker 6
15 – D#1	Africa 1

16 – E1	Africa 2
17 – F1	Africa 3
18 – F#1	Africa 4
19 – G1	Africa 5
20 – G#1	Metal
21 – A1	Slam
22 - A#1	Skin wood
23 – B1	China cymbal 1
24 – C2	China cymbal 2
25 - C#2	China cymbal 3
26 – D2	China cymbal 4
27 - D#2	Bad tambourine 1
28 – E2	Bad tambourine 2
29 – F2	Wood 1
30 - F#2	Wood 2
31 – G2	Indian 1
32 - G#2	Indian 2
33 – A2	Indian 3
34 - A#2	Indian 4
35 – B2	Djembe 1
36 – C3	Djembe 2
37 – C#3	Djembe 3
38 – D3	Bendil 1
39 – D#3	Bendil 2
40 – E3	Brazil 1
41 – F3	Brazil 2
42 – F#3	Chrismas bell 1
43 – G3	Chrismas bell 2
44 – G#3	Chrismas bell 3
45 – A3	Iron
46 – A#3	Hambox 1
47 – B3	Hambox 2
48 – C4	Mallet 1
49 – C#4	Mallet 2
50 – D4	Digit 1
51 – D#4	Digit 2
52 – E4	Zulu cow
53 – F4	Zulu bird
54 – F#4	Flextone
55 – G4	Plastic
Sound FXs	
56 – G#4	Shut down
57 – A4	Space fx
58 - A#4	Heart
59 – B4	Dtrain
60 – C5	Alien clap
61 - C#5	Whistling toad
62 – D5	Alien bird
63 - D#5	Crazy wire
64 – E5	Asian call
65 – F5	Ufo jam

66 - F#5	Lazer bass drum
67 – G5	Alert 1
68 - G#5	Alert 2
69 – A5	Alert 3
70 - A#5	Bubble
71 – B5	Storm
72 – C6	Busy
73 - C#6	Buzz
74 – D6	Electro beat
75 - D#6	Aowah
76 – E6	Space bubble
77 – F6	Electro
78 - F#6	Lazer gun 1
79 – G6	Lazer gun 2
80 - G#6	Rotor 1
81 – A6	Rotor 2
82 - A#6	Contact
83 – B6	Virtual claps

BKDBT44 special variation instruments

Bkdbt44 Pads:

PC	C0	VARIATION	COMMENT
120	1	Space pad	
120	2	Ice pad	
120	3	Dream pad 1	
120	4	Dream pad 2	Smoother than Dream pad 1
120	5	Angel pad	
120	6	Spirit choir	
120	7	Gospel choir	
120	8	Lyle pad	
120	9	Glass pad	
120	10	Cool pad	
120	11	Cyber pad	
120	12	Pray pad	
120	13	Fly voice	
120	14	Sunset	
120	15	Ober pad	
120	16	Sky pad	
120	17	Bach pad	
120	18	Wave pad	
120	19	Lunar pad	May also be used as lead
120	20	River pad	
120	21	Volcano	
120	22	Snow pad	
120	23	Cloud pad	
120	24	Jah pad	With 5th
120	25	Air pad	
120	26	Mystery	
120	27	Pink pad	
120	28	Obpad	
120	29	Winpad	

Bkdbt44 Basses:

PC	C0	VARIATION	COMMENT
121	1	Techno bass 1	Arpeggiator standard
121	2	Techno bass 2	Square
121	3	Techno bass 3	Saw
121	4	Sub bass	Sub
121	5	Tiny bass	
121	6	Dream bass	No decay
121	7	Young bass	Sub
121	8	Amazing bass	Random filter attack
121	9	Sequenced	Octaver
121	10	Sub fifth	With 5th
121	11	Double subbass	Sub with octaver
121	12	Hiphop bass 1	Close to GM synth bass
121	13	Hiphop bass 2	Fat sub
121	14	Hiphop bass 3	Dirty
121	15	Fonk bass	Wide
121	16	80's bass	Smooth
121	17	Lowest bass	Real da sub bass
121	18	West coast bass 1	Sub with filter
121	19	West coast bass 2	Sub with filter
121	20	Dynamic bass	No decay
121	21	Space slap	Sub slap bass
121	22	Live bass	Sub sine
121	23	Subsequence	Sub
121	24	Jungle bass 1	Sub
121	25	Jungle bass 2	Sub
121	26	Jungle bass 3	Sub
121	27	Jungle bass 4	Sub
121	28	Jungle bass 5	Sub
121	29	Jungle bass 6	Sub
121	30	King bass	Sub
121	31	Ion bass	Sub
121	32	Train bass	Sub
121	33	Smart bass	Sub

Bkdbt44 Leads:

PC	C0	VARIATION	COMMENT
122	1	'58 Lead	
122	2	Techno keyboard 1	
122	3	Techno keyboard 2	
122	4	Techno lead 1	With 5th
122	5	Techno lead 2	With 5th and filter
122	6	Techno lead 3	
122	7	Techno fat 1	Fat analog
122	8	Techno fat 1	Fat analog with bright attack
122	9	Techno fat 1	Fat analog with octave down
122	10	Synth piano	
122	11	Synth chord 1	Distortion and no decay
122	12	Synth chord 2	With 5th and no decay
122	13	Analog lead 1	
122	14	Analog lead 2	
122	15	Analog lead 3	
122	16	Analog lead 4	Squizy
122	17	Analog lead 5	Nasal
122	18	Analog lead 6	Nasal
122	19	Warna	
122	20	Warna 2	Filtered
122	21	Warna 3	
122	22	Motor bike	
122	23	Cyclebike	No attack
122	24	Killer tone 1	
122	25	Killer tone 2	With distortion feedback
122	26	3rd minor	Sampled chord
122	27	3rd major	Sampled chord
122	28	Solo synth 1	
122	29	Solo synth 2	
122	30	Solo synth 3	
122	31	Solo synth 4	
122	32	Solo synth 5	
122	33	Solo synth 6	
122	34	Solo synth 7	
122	35	Solo synth 8	
122	36	Solo synth 9	
122	37	Solo synth 10	
122	38	Solo synth 11	With 5th
122	39	Solo synth 12	
122	40	Solo synth 13	
122	41	Solo synth 14	
122	42	Zaw' world	With 5th
122	43	Russ' world	May also be used as pad
122	44	Venus	May also be used as pad
122	45	Planet	May also be used as pad

Bkdbt44 FXs:

PC	C0	VARIATION	COMMENT
123	1	Sweep	Propeller
123	2	Siren	
123	3	Siren FX	Filtered siren
123	4	Siren set	Muliple siren
123	5	Mars	Random pitch
123	6	Scratch	Scratch noise
123	7	Crack	Vinyl noise
123	8	Hiphop fx	
123	9	Jungle bass fx 1	Long slide
123	10	Jungle bass fx 2	Short slide
123	11	Bass mute	Fx on release and sustain

Bkdbt44 Percussions:

PC	C0	VARIATION	COMMENT
124	1	Tambour	Tambourine with delay
124	2	Tribal	From a marimba sample
124	3	Native	Tam-tam with breath noise
124	4	Whisper	Maracas
124	5	Nepal tambourine	Tambourine and tam-tam
124	6	Ragga tom	Electric tom
124	7	Ragga fx	
124	8	Tribut	Tam-tam with hand noise
124	9	Wha wha	

SmartMedia™Card Technical Information

What is the SmartMedia™Card?
Each SmartMedia™Card contains a built-in semiconductor memory chip, which is used for storing data.

Protecting you data:
In the situations listed below, data recorded on a SmartMedia™Card can be unintentionally erased.

- When the SmartMedia Card is used incorrectly.
- If the Card is exposed to static electricity or electrical noise.
- If the SmartMedia Card is removed during data recording, card formatting, or at any time other than as detailed in the MadPlayer instructions.

Handling the SmartMedia Card:

- Do not bend, drop or expose the Card to excessive shocks.
- Do not store or use the card where it may encounter strong static electricity or electrical noise.
- Do not store or use the Card in very humid or hot environments.
- Do not touch the Card's contact area, and do not allow the area to become dirty. Use a dry cloth to wipe away dirt, if needed.
- Keep Card in static-free case when not in use.
- Do not carry Card in pockets where it can be bent or sat on.
- Card may feel warm when first removed from MadPlayer. This is normal and does not indicate a problem.
- Affix Card labels only as directed by manufacturer.

MadPlayer Specifications

Physical

Size:	170 mm x 100 mm x 35 mm.
Weight:	100g without batteries.

Audio out
Female 3.5mm jack, stereo.
Audio output compatible with Headphones, or line-in connection.

Microphone in
Female 3.5mm jack, mono.
Microphone input compatible with electret condenser and dynamic microphones.

SmartMedia™Card slot
Compatible with 4MB, 8MB, 16MB and 32MB and 64MB SmartMedia™ memory cards.
Compatible with SmartMedia™ cards formatted as per SSFDC standard.

Power
2 "AA" batteries, or 2 rechargeable "AA" batteries

This page intentionally left blank.

USING THE MICROPHONE

- Select Music Highway Mic Lane: use joystick right until in Mic Lane.
- Change Voice Pitch: hold Pitch/Tempo + move joystick up/down.
- Add Echo Effects: hold FX key + move joystick left/right.
- Control Mic Volume: hold FX key + move joystick up/down.
- Mute Mic: press ■ *stop*.
- Un-Mute: press ▶ *play*.

QUICK COMMANDS

Open Music Style Menu..... ♫ *e-DJ*
Select Music Style ..... Up/Down.
Play New Song ..... ▶ *play*.
Start Another Song ..... ▶▶ *forward*.
Replay Current Song ..... ◀◀ *reverse*.
Save Current Song ..... *save/edit*.
Enter Tunnel Lane ..... Joystick down.
Return to Music Highway... Joystick up.

LISTEN TO THIS!

MadPlayer Demo Songs. Hear what people have created with the MadPlayer.

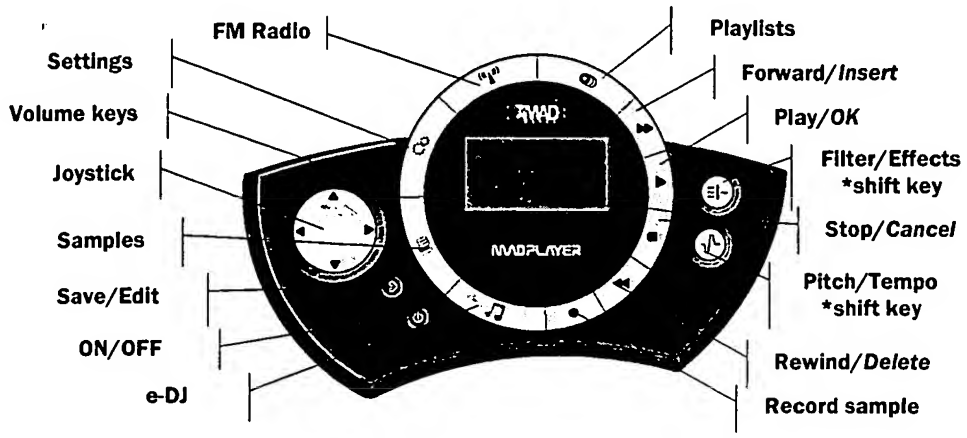
- Press *Songs*.
- Select "Playlists" Category—joystick up/down, then press ▶ *play*.
- Select DEMO—joystick up/down, then press ▶ *play*.
- Press ▶▶ *forward* to hear next song.
- Press ◀◀ *reverse* to hear previous song.

MAKE CONTACT

The MadPlayer does *so much more* than what fits on this card. To see the full manual for MadPlayer visit our website:

www.MadPlayer.com

For Technical Support
N. America (Hours 9-5 EST) :
1-866-MAD-PLAY (or 1-866-623-7529)
Or e-mail us at support@madwaves.com



*Shift keys must be held down while using joystick left/right and up/down.

MADPLAYER™ PACKAGE CONTENTS

- The MadPlayer™
- Headset + Embedded Microphone
- 32 MB SmartMedia™ Card
- USB Cable
- 2 "AA" Rechargeable Batteries & Charger
- MadQuickstart™ Card
- CD-ROM with MadWare™ Software, Samples, MadManual™, etc.

MADPLAYER™ SETUP

1. Install SmartMedia™ Card

If card is already in unit, press gently to make sure it is fully inserted. If card is not in unit, remove card from its holder and follow instructions below.

Card Loading

The best time to insert the SmartMedia™ Card is before you turn on the MadPlayer™.

Turn over the MadPlayer™. Hold the card so that its gold-colored contacts face the back of the MadPlayer™, and so that the card end with a notched corner is the end that will be fully inserted. Now slide the card into the slot as far as it will go. [The card will not slide all the way into slot if inserted facing the wrong way. Do not apply force to the card. It should slide in easily.]

Card Removal - *Important Notice!*

The MadPlayer™ *must* be turned off before you remove a SmartMedia™ Card.

2. Insert Batteries

Press, then slide open compartment door. Insert two AA alkaline, lithium or rechargeable batteries.

3. Plug in Headset/Microphone

Plug the two color-coded jacks into same-color jacks on top right of MadPlayer™ - red for Microphone input, and black for Headset. MadPlayer™ can also be connected to most external speaker systems, or microphones, using these same two jacks.

Turn over Card to see MadPlayer™ Key Functions

Now Grab Your MadQuickstart™ Guide
Turn on Your MadPlayer™
Make Music !

TURN IT ON

Press the on/off (⏻) button for 2 seconds. This will start playing the demo playlist.

- To hear next demo song: press ⏩.
- To hear previous demo song: press ⏪.

MAKE MUSIC

1. Press the *e-DJ* (♫) button.
2. Select Music Style: joystick up/down.
3. Start the music: press ▶ *play*.

MadPlayer™ begins composing a new song while you are listening.

- Want a different song, same style: press ⏩ *forward*.
- Want to change style: press ♫ *e-DJ*.

SAVE IT...PLAY IT!

- Save the Current Song: press ⬖ *Save/Edit* briefly *before* song ends.
- Play a Saved Song: press 💿 *Songs*.
- Select "Songs" category: press ▶ *play*.
- Scroll Song Titles: joystick up/down.
- Play Selected Song Title: press ▶ *play*.

ON THE HIGHWAY

Once a MadSong begins, you're on the 6-Lane Music Highway: Drums, Bass, Riff, Lead, Sample and Mic.

- Select a Lane: joystick left/right.
- Change Lane Volume: hold ⫶|- FX key+move joystick up/down.
- Change Lane Sound: hold ⫶|- FX key+move joystick left/right.
- Solo Lane Instrument: press ▶ *play* for two seconds.
- Un-Solo Lane Instrument: press ▶ *play* for two seconds.
- Mute the Lane: press ■ *stop*.
- Un-Mute the Lane: press ■ *stop* again.
- Pause Song: press ▶ *play*.
- Resume: press ▶ *play* again.
- Change Song Pitch: hold √- Pitch/Tempo + joystick up/down.
- Change Song Tempo: hold √- Pitch/Tempo + joystick left/right.

FUN IN THE TUNNEL

You can "Tunnel" under any Music Lane to change what you hear.
Song sections will repeat until you return to the Music Highway.

- Enter the Tunnel of a Highway Lane: joystick down.
- Change Instrument Pattern: joystick right.
- Previous Instrument Pattern: joystick left.
- Change Instrument: press ⏩ *forward*.
- Previous Instrument: press ⏪ *reverse*.
- Change the Sound:hold ⫶|- FX key + move joystick up/down/left/right.
- Solo an Instrument: press ▶ *play*. Un-Solo: press ▶ *play* again.
- Mute an Instrument: press ■ *stop*. Un-Mute: press ■ *stop* again.
- Return to Music Highway Lane: joystick up.

(To enter a different Tunnel, you must first return to Music Highway: joystick up. Select a different Highway Lane: joystick left/right. Then enter that Tunnel: joystick down.)

RECORDING SAMPLES

- Record a Microphone/Music Sample: press ● *record*, in Highway or Tunnel.
- Stop Recording Sample: press ● *record* again.
- Play *your* Sample: press ▶ *play*.

SAMPLING ON THE HIGHWAY

- Select Sample Lane: joystick left/right.
- Hear a Sample: press ▶ *play*.
- Change Sample: hold ⫶|- FX key + move joystick left/right.

SAMPLING IN THE TUNNEL

- Change/Play Sample: joystick left/right, then press ▶ *play*.
- Select Sound Effect: hold ⫶|- FX key + move joystick left/right.

dBTech

Custom graphics specification

Table of content

1 Introduction

This document describes the way graphic screen customization is implemented in the dB1 product. It describes how the custom graphs are stored on the SmartMedia card (custom graphs files), the mechanism used to load custom graphs in RAM and display them, as well as the parameters supported by custom graphs (Applicable mode, size, location, animation...).
This document serves as a basis to implement firmware code managing the custom graphs as well as the PC software conversion utility to import images in the custom graphs file format.

2 Custom screen feature overview

The custom screen feature allows the user to create a specific graphic screen associated to a given operating mode (or menu) of the dB1.
There must be an easy way to create those graphics either with a specific editor or by importing images from a standard format (BMP, DIB, GIF...) via a conversion utility.
The graphics are stored in a dB1 system file (slotted file format) on the SmartMedia card. This gives more flexibility to change the custom graphs, the drawback being that custom graphs are not supported if no SmartMedia card is present.
The choice between the default and the custom graphics for a given mode is made automatically by the dB1 when entering this mode (or menu). If a custom graph file is present on the SmartMedia card, the dB1 will display the custom graph. The default graph will be displayed in all other cases. This relieves the user from the burden of configuring an option in a system menu.
The custom graph file format and the display mechanism must ensure backward compatibility between different dB1 firmware releases and different versions of custom graphs files. A new firmware must operate with custom graphics from all versions prior to the release of the firmware. Conversely, an old firmware operates with custom graphics of a more recent version.
This is achieved by the specific format of the custom graphs file that groups parameters in specific slots. When new features are added, a new slot type is created. A firmware ignores information in a slot of unknown type.

Images forming a custom graph must comply to the following constraints in release 0.0 of the custom graph implementation.:

- The images are monochrome (black/white)
- The images have a fixed size of 128x56 pixels.

NOTE: in the release 0.0 of the custom graphs feature, only the Home screen supports custom graphs.
A graph can consist of a single image or of several images with a specified animation (looped, one shot, number of images per second...).

3 Loading and displaying graphs

The mechanism used to display the custom graphs is a multi step process started when entering a mode supporting custom graphics. The main phases of the process are described below:

When entering the mode:

1. The routine handling the screen will scan the cli_table[] in RAM to find out if a custom graphs file exist on the SM card (a file with the required type (SYSTEM) and sub-type. If more than one file is present, the first one in the cli_table[] is chosen.
2. If no such file exists, the default screen is displayed from the flash memory. If a file exists, the parameters and images are loaded in RAM after verification of the compatibility with the current platform and a flag is raised to indicate that custom graph must be used.
3. The first image is displayed and the graphic timer is loaded according to the animation parameters found in the file.

Upon subsequent occurrences of the graphic timer in this mode:

1 If the custom graph flag is raised and the graph is animated, a new image is displayed from RAM and the current image index is incremented.

2 When all images of the graph have been displayed, the animation stops or starts again with the first image if the type is looped.

4 Custom graphs file format

The custom graphs are stored in a file in the "slotted" format used for all dB1 proprietary files. See Appendix A for a description of the slotted file format. This file is of dB1 system file type (SYSTEM_FT) with graphs subtype. The file has a .DBS extension.
A single file contains all graphic information for all screens supporting custom graphs.
A screen (usually corresponding to a given operating mode of the dB1 – e.g. Home) is described by a set of slots called a **screen block**. There are as many screen blocks as customized screens.
A **screen list slot** located after the SLS header, lists all screen blocks and their offset relative to the start of the file.
A valid custom graphs file contains at least one SLS header, on screen list slot and one screen block.

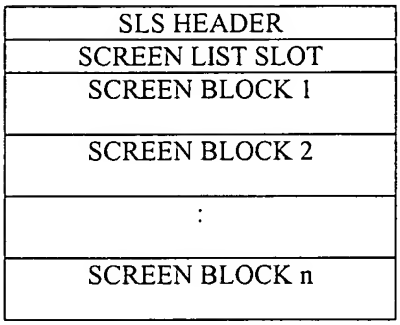


Figure 1: Custom graphics file format

NOTE: For a description of the generic slotted file format, refer to Appendix A.

4.1 SLS Header

The Slotted file Header, SLS HEADER contains the following specific values for custom graphs files:
File Type: SYSTEM_FT + GRAPHICS_ST (see db1dynmem.h)
SLS type: CUSTOM_GR_SLS
SLS version: current version (Initial version is 0x0000)
Data length field length: 2
Number of slots: depends on number of images. Usually equals number of images plus 2 if no image parameter slots are used.

4.2 Screen list slot

The screen list slot follows the SLS header and lists all screen blocks present in the custom graphs file. Each screen block is identified by the screen type it applies to and the offset of its screen block from start of file.
It is used to find rapidly the corresponding screen block when entering a new operating mode.

Slot type: GR_FL_LIST_SL
Slot name:

Name length: 0
Name: None

Slot data:

Data length: 5 x n (n number of custom screens in file).

Data:

Parameter	Size in bytes	Description
Screen 1 type	1	DB1 screen type to which this screen applies. Value matches the db1 enum for this screen type (e.g. DBS_HOME).
Screen 1 offset	4	Offset of first byte of screen block from start of file.
Screen 2 type	1	DB1 screen type to which this screen applies. Value matches the db1 enum for this screen type (e.g. DBS_HOME).
Screen 2 offset	4	Offset of first byte of screen block from start of file.
:		
Screen n type	1	DB1 screen type to which this screen applies. Value matches the db1 enum for this screen type (e.g. DBS_HOME).
Screen n offset	4	Offset of first byte of screen block from start of file.

4.3 Screen block

A screen block is made of one **screen parameter slot**, one or more **extended screen parameter slots** and one or several **image data slots** and optional **image parameter slots** and **extended image parameter slots**.
A screen block consists in at least one screen parameter slot and one image data slot.
Depending on the SLS version, zero or more extended screen or image parameter slots can exist. To guaranty backward compatibility, a parameter slot of a given type and version will contain all the parameters defined for the same type in previous versions in the same order and starting from the first parameter.

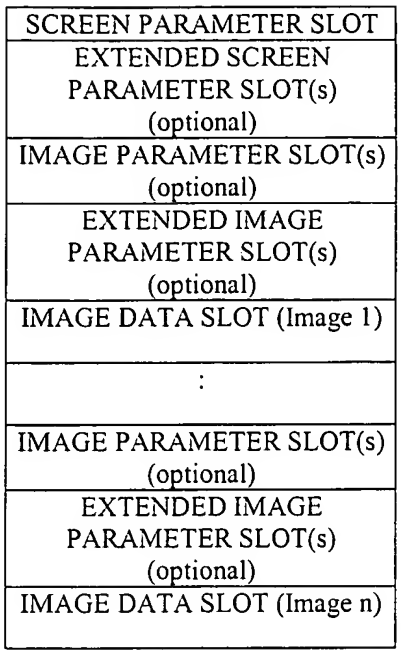


Figure 2: Screen Block format

4.3.1 Screen Parameter slot

The screen parameter slot contains basic information describing the graphic screen. If more specific parameters are required for a given screen type, one ore more extended parameter slots will be used (TBD).

Slot type: GR_SC_PARM_SL
Slot name:
Name length: 0
Name: None

Slot data:
Data length: 8
Data:

Parameter	Size in bytes	Description
Screen type	1	DB1 screen type to which this screen applies. Value matches the db1 enum for this screen type (e.g. DBS_HOME).
Num images	1	Number of images to follow. Max 255.
Animation	1	Type of animation: SCA_SINGLE(0): The images are displayed in the order of the image slots once and the animation stops. This is the default value if only one image is present SCA_LOOP(1): The images are displayed in the order of the image slots and back again from the start continuously. n (2-254): The images are displayed in the order of the image slots and back again for a number of loop equal to n. SCA_SPECIAL(255): Animation parameters are described in a specific extended parameter slot (TBD).
FPS_2	1	Twice the number frames per seconds. This value is used if the animation is not of type SCA_SPECIAL and indicates the number of

		images displayed per second. Values range from 0 for a static image to 20 for 10 images per second.
Width	1	Width of the graph in pixels. This is the default width for the graph, meaning that if an image has no image parameter block associated, this value is used as the image width. Values range from 1 to 128.
Height	1	Height of the graph in pixels. This is the default height for the graph, meaning that if an image has no image parameter block associated, this value is used as the image height. Values are multiples of 8 and range from 8 to 56.
Column	1	Default X coordinate of the bottom left pixel of images in this graph. If an image has no image parameter block associated, this value is used as the image base X coordinate. Value range from 0 (1st column at left of LCD) to 127.
Line	1	Default Y coordinate of the bottom left pixel of images in this graph. If an image has no image parameter block associated, this value is used as the image base Y coordinate. Values range from 15 (16th line from top of LCD) to 63. Values start at 15 because status line (8 pixels) is excluded from custom graphs and the image height is at least 8 pixels.

4.3.2 Extended Screen Parameter slot

The extended screen parameter slot is optional and contains specific screen attributes for the screen like advanced animation definitions or special instructions for image layout that may be required for certain type of screens.
The extended screen parameter slot is inserted just after the screen parameter slot.
The slot type is composed of a base slot type (GR_EXT_SC_PARM_SL) on MSB and a version on LSB. This permits to regroup all extended image parameters in a single slot. The version field allows to identify which parameters are present in the slot. A new version must include all parameters of all previous versions in the same order to guaranty backward compatibility.

NO application defined yet .

4.3.3 Image Parameter slot

The parameter slot is optional and contains image attributes for the image contained in the following image data slot.
The slot type is composed of a base slot type (GR_IM_PARM_SL) on MSB and a version on LSB. This permits to regroup all extended image parameters in a single slot. The version field allows to identify which parameters are present in the slot. A new version must include all parameters of all previous versions in the same order to guaranty backward compatibility.

Slot type: GR_IM_PARM_SL+1
Slot name:
Name length: 0
Name: None

Slot data:
Data length: 5
Data:

Parameter	Size in bytes	Description
Width	1	Width of the image in pixels. Values range from 1 to 128.
Height	1	Height of the image in pixels. Values are multiples of 8 and range from 8 to 56.
Column	1	X coordinate of the bottom left pixel of the image. Value range from 0 (1st column at left of LCD) to 127.
Line	1	Y coordinate of the bottom left pixel of the image. Values range from 15 (16th line from top of LCD) to 63. Values start at 15 because status line (8 pixels) is excluded from custom graphs and the image height is at least 8 pixels.
Repeat	1	Number of times the image must be repeated in case of animation. (1 to 255.

4.3.4 Extended Image Parameter slot

The extended image parameter slot is optional and contains specific image attributes for the image contained in the following image data slot. If one image parameter slot is present, the extended parameter is inserted just after it and before the image data slot.
The slot type is composed of a base slot type (GR_EXT_IM_PARM_SL) on MSB and a version on LSB. This permits to regroup all extended image parameters in a single slot. The version field allows to identify which parameters are present in the slot. A new version must include all parameters of all previous versions in the same order to guaranty backward compatibility.

NO application defined yet .

4.3.5 Image Data slot

The image data slot contains image name and data describing the image.

Slot type: GR_IM_DATA_SL
Slot name:
Name length: n (Max 255).
Name: Name of the original image file from which the image has been converted. Not used by firmware but maybe needed to trace back image origin.

Slot data:

Data length:	Image Width * Image Height / 8
Data:	Image data formatted as expected by display_graph() routine . Each byte contains the pixels of 8 lines of one column of the current page. Bytes are ordered starting from bottom left corner of the image, going left until the last column of the image and then up to next page (group of 8 lines) of the image. The logic behind this is to ease conversion between bitmap files and graph data files. Bitmap files store data line after line, starting from bottom line. Pixels for each line are stored from left to right. So the conversion from bitmap to graph files builds an array starting with the first 8 lines of the bitmap (8 bottom lines) and so on up to last page.

Appendix A: GENERIC SLOTTED FILE FORMAT

This format is common to the following kinds of Slotted Files:
- S List Files (i.e. Song Lists and Sample Lists);
- the System Parameters File;
- the File Settings File;
- the Radio Presets File;
- the User Song Files;
- the Sample Files.
- the Custom graph Files.

Note 1: in all the Slotted Files, whenever a field is used to store a numerical value on more than one byte, "little endian" format is used (i.e. the first byte of the field holds the least significant byte, and the last byte of the field holds the most significant byte).

Note 2: in all the Slotted Files, "lengthes" are expressed in bytes.

Slotted Structure
A Slotted Structure (SLS) is made of a Slotted Structure Header followed by N Slots, where each Slot contains an item:

SLS Header	Slot 0	Slot 1	...	Slot N-1

The SLS Header

The SLS Header has the following format:

Header Length (14)	Checksum	File Type	SLS Type	SLS Version	Data Length Length (n1)	Num Slots (N)
2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes

The Header Length field contains the length of the Header. It is used to allow future expansion of the Header without creating an incompatibility.

WARNING: the Header Length field contains the length of the Header including itself (in the current Release, the length of the Header is 14). This is unlike the length fields of a Slot (Name Length and Data Length), which contain respectively the length of the Name field alone and the length of the Data field alone.

The Checksum field contains the checksum of the Slotted File. The checksum is computed by adding all the bytes of the File modulo 65536, not including the checksum field itself (for a detailed description see "File Checksum computation" in write_file_type_and_checksum()).

The File Type field contains the File Type (Base Type + sub-type) of the Slotted File. It used when the sub-type is not null to determine the exact File Type when the File is imported, since the File Extension only determines the Base File Type. For consistency, this field is written whenever a Proprietary File is created or modified, even if it is of a type whose sub-type is always null.

The SLS Type field indicates what kind of Slotted File this is, i.e. whether this is a Song List, a Sample List, a File Settings File, etc. This information can also be derived from the File Type of the Slotted File (Note 2), but storing this information in the Slotted Structure itself (i.e. in the Slotted File data) permits to have complete information about the Slotted Structure independently from the File that contains it (Note 3).

The SLS Version field is used to maintain compatibility if the format of a kind of Slotted File changes. Song List Files and Sample List Files (S Lists) share the same Version number (since they are vewry close to each other, they will evolve together) but all the other kinds of Slotted Files have distinct Version numbers (presently they all have the same numerical value, but this is liable to change)).

The high byte of the Version field is the Base Version number and the low byte of the Version field is the sub-Version number. For instance, Version 2.5 is stored as 0x05 0x02, in that order (if decoded as a word value, this would yield 0x0205).

The Data Length Length field contains the number of bytes that the Data Length field of each Slot contains (refer to the description of the Slot format below). It is used to optimize the size of the Slotted Files, because certains types of Slotted Files use a very small Data field or no Data field at all (e.g. Sample and Song Lists), whereas other types of Slotted Files use a very big Data field (e.g. Samples).

The Num Slots field holds the number of Slots (N) in the List (Notes 4 and 5).

The Slot format

Each Slot has the following format:

Slot Type	Name Length (n2)	Name	Data Length (n3)	Data
2 bytes	1 bytes	n2 bytes	nl bytes	n3 bytes

where nl = Data Length Length value in SLS Header

- the "Slot Type" field indicates the nature of the contents of the Slot in case different kinds of Slots are used in a given Slotted File (Note 8). In the special case of File Slots, i.e. Slots whose contents are a reference to a File, by convention, the "Slot Type" holds the File Type of the referenced File. This is the case in S Lists and in the File Settings File (Note 9).

- the "Name Length" field holds the length of "Name" in bytes. If no Name is present, "Name Length" = 0;

- in the case of File Slots, the "Name" field holds the name of the referenced File (S Lists, File Settings File) (Note 6), else it can hold some other name (like the Name of a Preset in the case of the Radio Presets File), or it can be left unused (in the latter case, it is absent, and "Name Length" = 0). The null string terminator is NOT included;

- "Data Length" holds the length of "Data" field in bytes. If no Data field is present, "Data Length" = 0 (Note 7);

- the "Data" field holds complementary information whose meaning depends on the type of Slotted Structure:
  * in Sample Lists and Song Lists, it is not used presently;
  * in the File Settings File, it holds the File Settings (n2 = FILE_STG_LEN);
  * in the Radio Presets File, it holds the Station Frequency (n2 = 2 because the Frequency is stored on a word);
  * in the System Parameters File, it holds the values of the Parameters of the relevant Parameter Group;
  * in Sample Files, in the Sample Data Slot, it holds the Sample data.

Note that the meaning of each field in a Slot may depend on both the type of Slot and the type of Slotted Structure.

File Slots

A special class of Slots are the Slots that hold a reference to a File. They are notably used in the S Lists and in the File Settings File (see the formats of these Lists below).

The Name field of a File Slot always contains the name of the File that the Slot references. The type of the File (from which its extension can be derived) can in some cases be derived from the type of Slotted Structure (i.e., the SLS Type). However, for clarity I recommend that the Slot Type of all File Slots, no matter in what type of Slotted Structure they are used, be equal to the File Type of the File that the Slot references. This makes it possible to state that an unambiguous reference to the File can be derived from the contents of the Slot, without resorting to the SLS Type.

The Data field of File Slots is currently empty, but it could in the future hold complementary information.

Alien Slots

"Alien Slots" (in a given type of Slotted File) are Slots whose type is not known in the current Release. They may exist if we read on Release A an SM Card written in Release B (where Release B is typically more recent than Release A).

As a general rule, all the Slotted Files must be able to accept Alien Slots, no matter where they are placed in the Slotted Structure, without creating an incompatibility. This incurs two constraints in the Slotted File management:
1 - Alien Slots must be ignored (they must have no effect in the current Release);
2 - Alien Slots must be preserved when the Slotted File is modified, i.e. they must not be deleted.

This will permit to add complementary information in any Slotted File in future Releases without creating an incompatibility: older Releases will be able to read the Slotted Files created in the new Release.

Note 3: the SLS Type (_SLS values) is often, but not necessarily, equal to the File Type (_FT values). Among other possibilities, the SLS Type would be useful if we needed to store a Slotted Structure

in the Slot of a higher level Slotted Structure. As an example, we might want (who knows) to store a mixture of Song Lists and Sample Lists in a single File (whose File Type would have to be specific, i.e. neither the File Type of a Song List File, neither that of a Sample List File).

Note 4: presently, no kind of Slotted Structure can actually exceed 256 Slots, so a byte would be sufficient to hold the number of Slots. However, using a word ensures compatibility can be maintained if we ever need to exceed that limit in the future.

Note 5: the number of Slots could be derived by parsing the Slotted Structure (with the end of the Slotted Structure given by the file length), but decoding is easier if the number of Slots is readily available. In fact, this redundant information is used to verify the consistency of the Slotted Structure (see verify_slotted_file()).

Note 6: when a Slotted File is used to reference a File, the File Type and the File Name of this File are stored in a Slot, to make the Slotted File relatively independent of the particular SM Card on which it is being used. If we stored the File Indexes, rather than the File Types and Names, any change in the Files (adding, deleting or renaming Files even if they are not part of that Slotted File) would modify the File Indexes (since File Indexes are in the sorting order). Storing the Cell indexes instead of the File Indexes would not help when a File is modified, because its Cell index changes in that case.

Furthermore, in the case of Song or Sample Lists, the Song or Sample List File would be closely tied to the SM Card in which it was created. It would not be possible, for instance, to copy a Song List File and the associated Songs from a Card to another because we cannot guarantee that the copied Songs would have the same Cell indexes in the destination Card. Storing the File Types and Names eliminates these problems and also makes it easy to edit a Song List File or Sample List File on a PC independently of a particular SM Card.

Note 7: the Data Length field has been defined on 4 bytes in order to remove any restriction on what the Data field can store. Presently, this large capacity is used to store the Sample data (i.e. the PCM or ADPCM data). In the other types of Slotted Files, the Data field is small or non-existent (a Data Length field would have been sufficient).

Note 8: the meaning of the Slot Types is not supposed to be global, i.e. the Slot Type value may need to be interpreted differently in each type of Slotted Structure. However, when creating a new type of Slot, we'll try to use a value which does not exist in another type of Slotted Structure, if this is not too much of a constraint.

Note 9: if a given type of Slotted File can hold File Slots, the other types of Slots should have Slot Types that do not conflict with File Types (we recommend using values <= RESERVED_SL_MAX).

MadWare Reference
(Preliminary)

1 MadWare™ Dashboard

The What, Weird, and How of the MadWare™ Dashboard:

What?
The MadWare™ Dashboard, Figure 1, is the control window that gives you access to all features of the MadWare software program. Through it you can communicate with the Madplayer™, transfer files between MadPlayer™ and your PC, and interact with the MadWaves web site.

Weird!
The Dashboard puts you in control of two distinct operating environments: MadPlayer Mode and MadWorld Mode.
MadPlayer Mode is the file transfer interface that allows you to move music files between a SmartMedia™ Card that's installed in your MadPlayer™, and the hard drive of your computer.
MadWorld Mode is the internet interface that allows you to upload/download music files to/from the MadWaves web site.
Note: When the MadWare™ Dashboard is first opened, it is always in MadPlayer Mode, for music file transfers.

How?
To open the MadWare™ Dashboard, either click the MadWare Icon on your desktop, or choose MadWare from the Programs list under the Start button.

Here's looking at you, Dashboard.
The central portion of the Dashboard window (Figure 1) is called the work area. Surrounding it is the Dashboard frame, which contains the direct-access connection buttons. In this shot the work area is divided into the two columns of MadPlayer Mode—the default mode for Dashboard. The right column displays a directory of all files that are on the SmartMedia™ Card of the MadPlayer™. The left column displays a directory of the contents of your computer's hard drive.

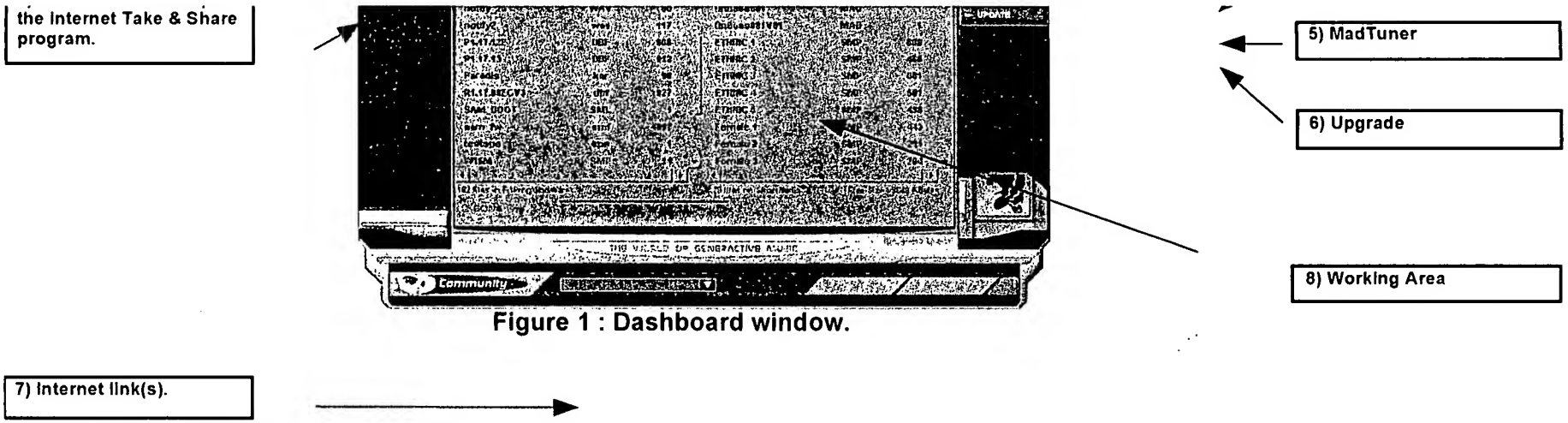


Figure 1 : Dashboard window.

Navigating the Dashboard:

MadPlayer Mode (1)
This is the File Transfer interface. And it is the default mode that the Dashboard assumes each time you open the MadWare™ program. Once the MadPlayer™ is connected to your computer with the MadWave™ USB cable, MadPlayer Mode allows you to move files between MadPlayer™ and your computer's hard drive. It also enables you to rename, delete and copy files, as well as edit their properties.

MadWorld Mode (2)
This is the Internet File Sharing interface, which connects you to the members section of the MadWaves web site. Once connected you can download/upload Song or Sample files, as well as access periodic upgrades for the MadPlayer™, or the MadWare™ software program.

Configuration (3)
Displays the version numbers for the Firmware and Sound Bank of a connected Madplayer™.

SmartMedia™ (4)
Displays the properties of the SmartMedia™ Card currently inserted in the MadPlayer™.

MadTuner (5)
Displays the MadTuner dialog window through which you can select and store Radio frequencies that can be listened to with the MadPlayer™ Radio. (See the MadManual™ for complete instructions for using the MadPlayer Radio.)

Upgrade (6)
Periodic upgrades of the MadPlayer™ Firmware and Sound Bank will be posted on the MadWaves.com web site. Those upgrade posting will provide current installation instructions which can be downloaded and printed. Following those instructions, and utilizing the Firmware and Sound Bank sections of the MadManual, you will use the Upgrade button of the dashboard to initiate the downloading and installation process.

Internet Links (7)
Connects you to the MadWaves.com web site.

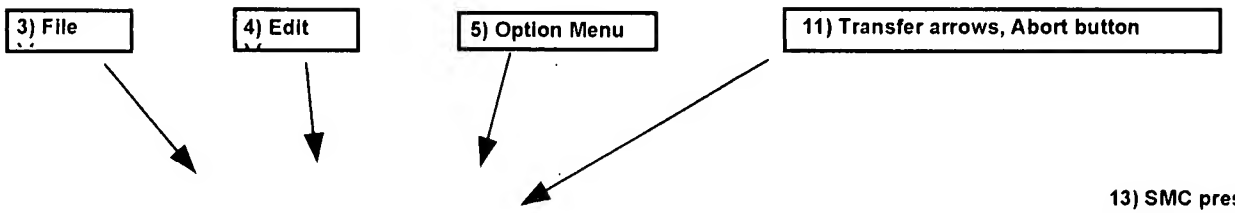
Work Area (8)
The central frame of the Dashboard window. In MadPlayer™ Mode it displays a two-column format. In MadWorld Mode it displays a succession of dialog boxes that allow you to initiate file trading through the MadWaves web site.

Help, About and Close (9)
Help opens the scrollable Help file for the MadWare™ software program.
About displays the version number and copyright for the MadWare™ program.
Close (X) exits the MadWare™ program.

2 MadPlayer Mode

MadPlayer Mode is all about moving files from your MadPlayer™ to your computer, and back again. It also enables you to copy, delete, rename or edit the properties of your music files.

When you open the MadWare™ software program, or when you press the MadPlayer button on the Dashboard, the two-column File Transfer interface appears in the work area of the Dashboard—Figure 2.



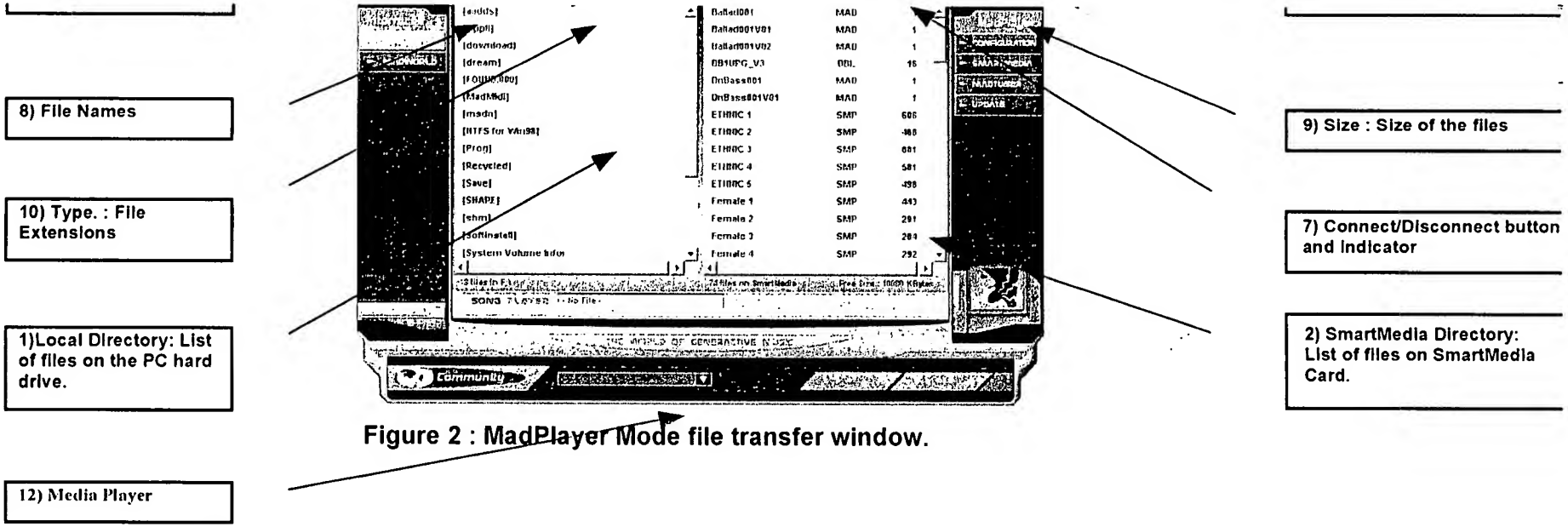


Figure 2 : MadPlayer Mode file transfer window.

Navigating the MadPlayer™ Mode File Transfer window:

Local Directory (1)
Displays a directory of the contents of your computer's hard drive.

SmartMedia™ Directory (2)
Displays a directory of all the files on the SmartMedia™ Card in the MadPlayer™.

Menu Buttons
File (3): Delete, Rename, Play and Convert and edit properties of files.
Edit (4): Create or Edit Playlists and Sample Sets.
Option (5): Opens the Filter dialog window to select file types that can be displayed in the Local and SmartMedia™ areas.

Browse button (6)
Opens a dialog window that allows you to select a different source for the Local Directory, such as a floppy or Hard drive. Note: The last directory used becomes the default directory when the program is closed and then restarted.

Connect/Disconnect (7)
This is both a button and an indicator. Click on the button to Connect or Disconnect MadWare™ from the MadPlayer™. Once

the operation is completed the button then reverts to the option not chosen. Meaning, while you are "Connected," the button will read "Disconnect." The text at the right of the button also indicates if a SmartMedia™ Card is inserted in the MadPlayer™.
MadWare™ must be in the disconnected state in order to play a song with the MadPlayer™, or for any other MadPlayer™ function.
Always disconnect MadWare™ before inserting or removing a SmartMedia™ Card. Disconnecting MadWare™ make the player return in Home Screen Mode so that you can insert or remove a SmartMedia™ Card.
MadPlayer™ must be turned on and in Home Mode before it is connected to your computer.

File Sorting Tabs
NAME (8): Click on NAME to sort the directory files in alphabetic ascendant or descendant order.

SIZE (9): Click on SIZE to sort the directory files in ascendant or descendant size.

TYPE (10): Click on TYPE to sort files by the file extension type.

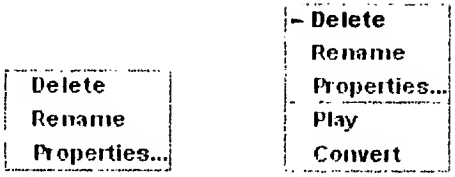
Transfer Arrows/Abort Button (11)
Highlight a file in either the Local or SmartMedia™ directory and then click on the appropriate arrow to copy that file to the other Directory. Press the Abort button if you want cancel the transfer.

Media Player (12)
The Media Player can play files stored on the Local Directory (audio & MIDI—but not MadSongs), and it also manages playlists if several files are selected in the Local Directory window.

Menus

Once a file is selected in either the Local or SmartMedia™ directory, a number of actions can be accessed through the File, Edit, and Options menu buttons. The functions available through each button are described below.

File Menu
Depending on the type of file that you have highlighted in one of the Directories, when you click on the File Menu you will open the selection box in either Figure 3a or 3b. (The File Menu can also be displayed by right-clicking on a selected file.)



a) b)

Figure 3 : a) standard "File" Menu
b) extended "File" Menu

Figure 3a
All files can be Deleted, Renamed or edited for Properties. The Delete and Rename functions mirror those used in basic windows file management. The editing of File Properties is explained below.
Delete shortcut: highlight file and press "Delete" key.
Rename shortcut: left click on already selected file.
Properties: These file properties can be edited: Name, size, duration, creation date, modification date, attributes (Read-only, Hidden).

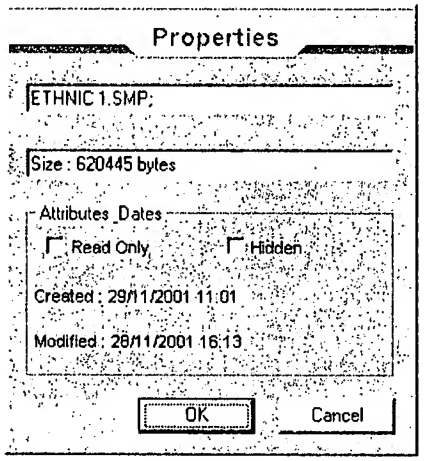


Figure 4 : File properties edition

The Properties screen also allows you to modify Read-Only and Hidden attributes.
If several files are selected, opening the Properties Edit screen will only display the total size of the selected files.

Figure 3b
The extended File Menu is only available when a single playable or convertible file is selected in the Local Directory.

Play—begin playing the selected file with the MadWare™ Player.
Note: Read this if there is no sound when playing MIDI files.
The MadWare™ Media Player uses the Windows default Midi Port (Selected by opening the Control Panel and choosing Sounds & Multimedia, then Devices, and then Midi Devices and Instruments). It also uses the default audio port to play audio files. The MadWare™ Media Player can only play MIDI files if "Mad MIDI Port" is NOT selected as the default Midi Port in the Sounds & Multimedia panel.

Convert: offers file conversion options for the selected file. Only files in the Local Directory can be converted.

Edit Menu
The Edit Menu (Figure 5) lets you create new Playlists and Sample Sets, or edit existing Playlists and Sample Sets. See Play Lists and Samples Sets for more information.

Figure 5 : Edit Menu.

Options Menu
The Options Menu (Figure 6) opens with the choose Filters box. Click on it to open the Filter selection window (Figure 7).

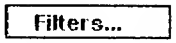


Figure 6 : Options menu.

The Filter selection window allows you to select the file types that will be displayed in both the Local and SmartMedia™ directories.

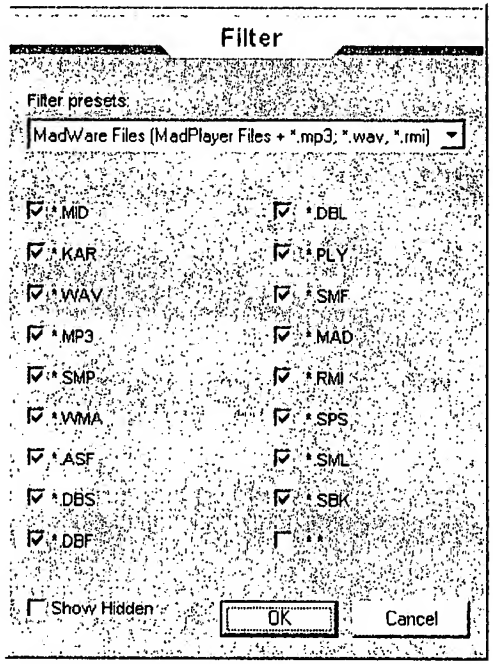


Figure 7 : Filter selection window.

The Filter selection window allows the following options for the display of file types:

- **MadPlayer Files—**all files that are recognized by the MadPlayer™ without conversion.
- **MadWare Files—**all of the above files, as well as the files that can be converted and then recognized by the MadPlayer™.
- **Any Specified Type**—a customized list of file types created with the check-off boxes.
- **All Files**

3 File Transfers

File Transfers are coordinated by MadWare™ while it is in MadPlayer Mode. The MadWare Dashboard window opens in MadPlayer Mode. If you are in MadWorld Mode, simply click on the MadPlayer™ direct access button and the Dashboard will revert to the two-column MadPlayer Mode work area.

Files can be transferred from the Local Directory to the SmartMedia™ Directory (left column/right column), or the reverse, using either of two simple procedures: The Transfer Arrows and Abort button of the Dashboard Transfer Area , or through standard Drag and Drop from one directory column to the other.

Transfer a file from the SmartMedia™ Directory to the PC

Each time a file (or group of files) is selected in the SmartMedia Directory (right column of work area), the right-to-left transfer arrow is highlighted. Clicking on the highlighted arrow initiates the file transfer process. The process can also be initiated by dragging a file from the SmartMedia Directory into the Local Directory. Files that are selected for transfer are *copied* to the target directory, while the original remains in the host directory.

During the transfer process, the Transfer Information bar (Figure 8) reports the status of the current operation—number of files to be transferred, number and name of file that is currently being transferred, speed of the transfer in Kbytes per second, etc.

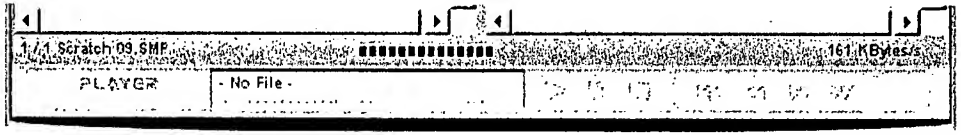


Figure 8 : Transfer Information bar.

Transfer a file from the PC to the SmartMedia™ Directory

Each time a file (or group of files) is selected in the Local Directory (left column of work area), the left-to-right transfer arrow is highlighted. Clicking on the highlighted arrow initiates the file transfer process. The process can also be initiated by dragging a file from the Local Directory into the SmartMedia Directory. Files that are selected for transfer are *copied* to the target directory, while the original remains in the host directory.

During the transfer process, the Transfer Information bar (Figure 8) reports the status of the current operation—number of files to be transferred, number and name of file that is currently being transferred, speed of the transfer in Kbytes per second, etc.

If you initiate the transfer of a file that is not directly supported by the MadPlayer™, a dialog box will appear and offer file conversion options. File conversion is done automatically for Midi and Karaoke files. See Conversions for more information.

4 File Conversions

Summary

a) When to use it : When you want to convert a file which format is not directly supported by the MadPlayer into a format supported by the MadPlayer

b) How to : There is 2 ways for a file to be converted. You can explicitly convert the file by clicking in the File/Convert menu or by right-clicking on the file you want to convert. Nevertheless, if you want to transfer, on your SmartMedia card, a file that is recognized by the MadWare but not by the MadPlayer (MP3, WAV, MIDI and Karaoke files in format 1), a conversion will be proposed to you before transferring.

c) Description :

4.1 WMA Conversion

The MadWare gives you the possibility to convert your WAV and MP3 files in WMA (Windows Media Audio) files. This format allows you to compress your files and play them on the MadPlayer. When you are going to convert a file in the WMA format, a pop-up will open:

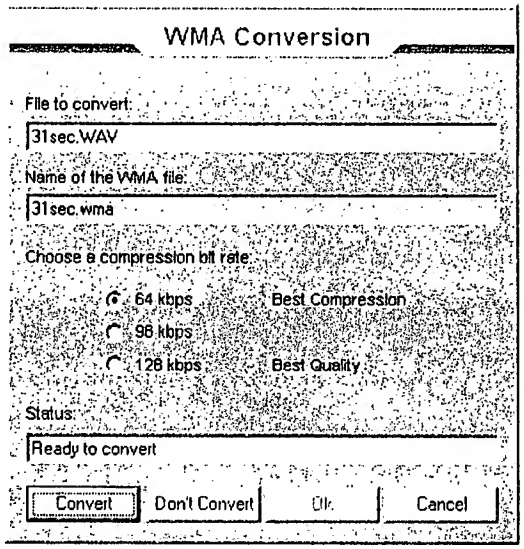


Figure 9 : WMA conversion dialog window

In this window, you can find several fields:

- File to convert: this is the name of the file you are going to convert. This original file will not be destroyed during the conversion.
- Name of the WMA file: this is the name of your future WMA file. The Madware propose you a unique default name but you can enter ever name you like. If the file already exists, you will be warned before overwriting.
- Compression bit rate: you can choose between 3 compression bit rates. More less is the compression, better is the quality. When you convert MP3 files to play on your MadPlayer, a value of 64 kbps should give you satisfactory results.
- Status: this status line gives you informations about the conversion progression. When the windows is opened, the status lines indicates "Ready to convert".

Actions :

- Convert: Press "Convert" to start your conversion when you have set the right name for you WMA output file.
- Don't Convert: Press this button if you want to transfer your file without converting it. If you are making a local conversion, "Don't Convert" has the same effect than "Cancel".
- Ok: Press on this button to transfer your file or to close the conversion window if you just made a local conversion.
- Cancel: Press this button to cancel the conversion at any time. You can cancel even if the status line indicates you that the conversion is already finished.

When you click on "convert", the conversion starts and the status line indicates "Converting". A sharp will be add every both seconds to show you the conversion progression.

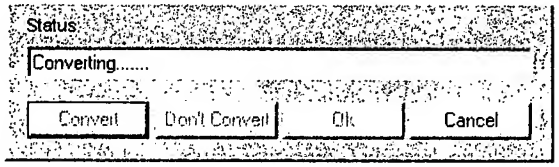


Figure 10 : Conversion is in progress

When the conversion is finished, the status line indicates "Conversion finished". Then, you can press Ok to begin the transfer (if you are transferring the file) or just to return in the Madware.

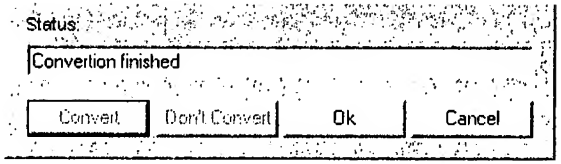


Figure 11 : Conversion is finished

4.2 SMP Conversion

The MadWare gives you the possibility to convert your WAV files in SMP (Sample) files. This format allows you to compress or not your files, convert and play them on the MadPlayer. When you are going to convert a file in the SMP format, a pop-up will open:

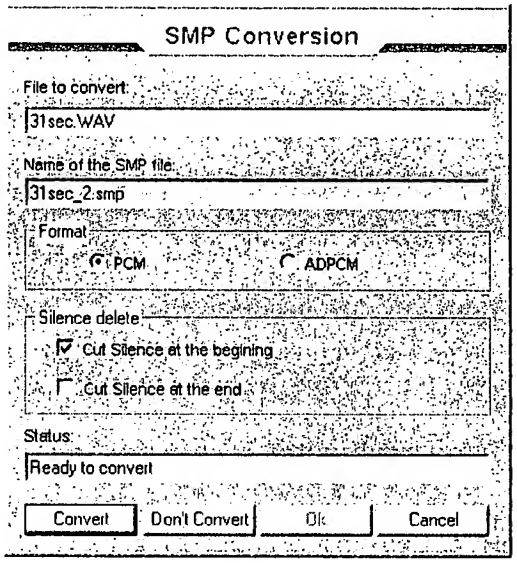


Figure 12 : SMP conversion dialog window

In this window, you can find severals fields:

- File to convert: this is the name of the file you are going to convert. This original file will not be destroyed during the conversion.
- Name of the SMP file: this is the name of your future SMP file. The Madware propose you a unique default name but you can enter ever name you like. If the file already exists, you will be warned before overwriting.
- Format: These two radio buttons allow you to make a choice for the format of your conversion. If you choose PCM, the sample will not be compressed during conversion. If you choose ADPCM, it will.
- Silence delete: Here you have two check boxes to cut silence at the beginning and/or at the end of your sample.
- Status: this status line gives you informations about the conversion progression. When the windows is opened, the status lines indicates "Ready to convert".

Actions

- Convert: Press "Convert" to start your conversion when you have set the right name for you WMA output file.
- Don't Convert: Press this button if you want to transfer your file without converting it. If you are making a local conversion, "Don't Convert" has the same effect than "Cancel".
- Ok: Press on this button to transfer your file or to close the conversion window if you just made a local conversion.
- Cancel: Press this button to cancel the conversion at any time. You can cancel even if the status line indicates you

that the conversion is already finished.

The conversion process is the same as for the WMA conversion. Progression is indicated by the status area, as described below.

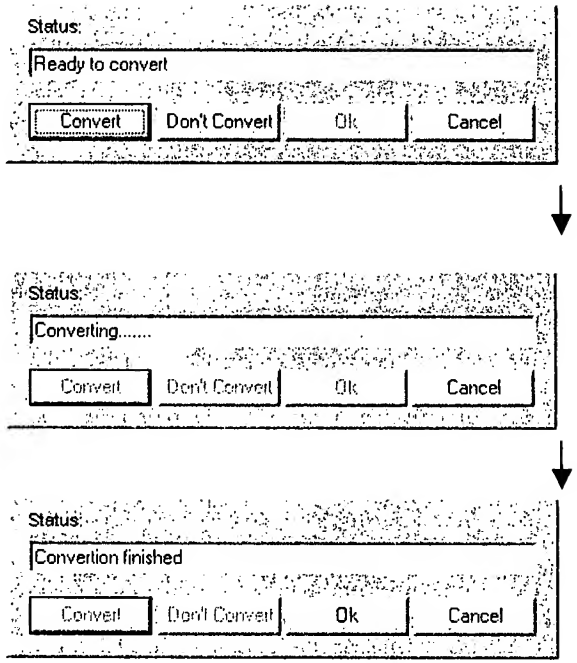


Figure 13 : Status information during conversion

4.3 MIDI Conversion

A MIDI file can be saved in 2 different formats: format 0 and format 1. The MadPlayer can only read MIDI file saved in format 0, then a conversion of your MIDI file could be required. If you transfer a format 1 MIDI file name on your SmartMedia, this will be converted but it is transparent for you: the name of the original file is kept and you have nothing to do.
You can of course convert your MIDI files without transferring them. In this case, a pop-up window will ask you for a new name :

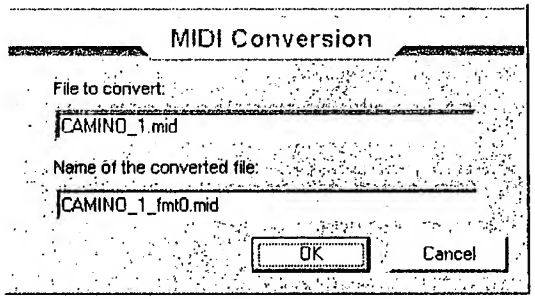


Figure 14 : MIDI conversion dialog window

4.4 KARAOKE Conversion

Karaoke files are MIDI files with some lyric informations. Then, all what was true for the MIDI conversion is true for Karaoke conversion.
Most of Karaoke files follow the MIDI Karaoke file type 1 format (Soft Karaoke) but there are several other formats that are a little bit different. The Madware conversion works well with the standard files but you can meet some problems with other formats. If the MadWare succeeds in converting a karaoke file, you will obtain a MIDI Karaoke file type 0. This format is compatible with the MadPlayer.
The standard extension for a karaoke file is .KAR but a file with the .MID extension can also be a karaoke file if it contains lyric informations.

5 Playlists and Sample Sets

With MadWare™ you can create, view and edit Playlists (file type .ply) and Sample Sets (file type .sps). For details about Playlists and Sample Sets please refer to the MadManual™ User's Guide.

The Edit Menu provides access to the dialog windows for Playlists and Sample Sets, Figure 15.

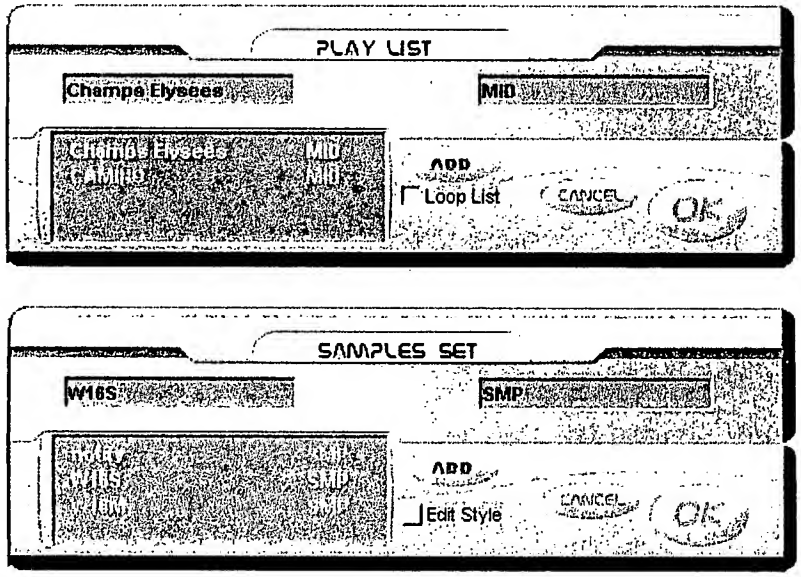


Figure 15: Playlist and Sample Set dialog window.

Creating/Editing Playlists and Sample Sets:

Add an item:
Click the Add button and select a file to be added.
Shortcut: Press the Insert (Ins) key.

Delete an item:
Select the item to be deleted, then press the Delete (Del) key.
Alternate: Right click on selected file and choose Delete.

Move an item:
Select the item to be moved, then use the Down Arrow or Up Arrow keys.
Alternate: Right click on selected file and choose Up or Down.

Loop a Playlist:
Click on the Loop List check box.

Choose a Sample Set Style:
Click the Edit Style button. A selection box opens that lets you choose from the following styles:
Dance, Hiphop, Cool, Techno, or Any Style.

6 Browser

The Browse button on the MadWare™ Dashboard opens the Browser window, Figure 16. This window allows you to select the computer directory that will appear in the Local Directory column of the MadPlayer Mode work area. This could be the directory of your hard drive, a floppy disk, a CDROM, or a specific folder in any of those locations. And it would be that directory to which/from which you could transfer files to/from the MadPlayer's SmartMedia™ Card.
Note : If you select the CDROM as the local directory, you will be able to read files from the CDROM, but not to write files to the CDROM (because it is a Read-Only device).

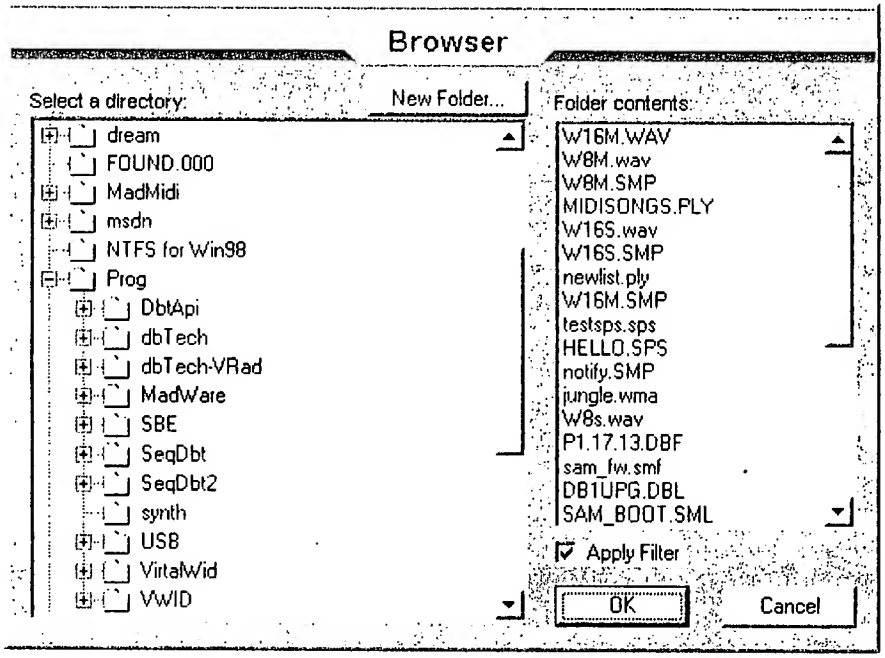


Figure 16 : Browser window.

Browser Window Functions:

Apply Filter:
When this box is checked, the Folder Content area only displays files that would appear in the Local Directory column of MadPlayer Mode, because of file types that were included or excluded by the Filters function. See Options Menu for more information about Filters.

New Folder:
This button creates a new folder in the directory that is open in the Select a Directory column.

7 MadPlayer Controls

These four Dashboard buttons supply information about, or access to, the connected MadPlayer™. See MadWare™ Dashboard.
The four MadPlayer Controls are:
Configuration : MadPlayer Configuration
SmartMedia :Smart Media properties
MadTuner :MadTuner
Upgrade : See the MadManual™ for details.

The function and use of each button is described below.

7.1 MadPlayer Configuration

The Configuration window supplies information about the Firmware and Sound Bank of the connected MadPlayer™.

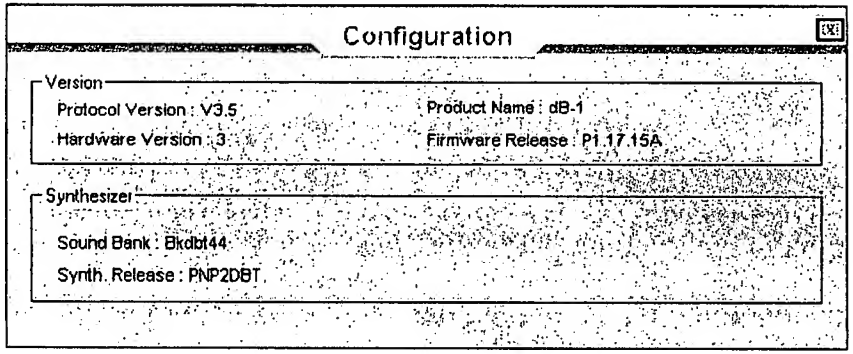


Figure 17 : Configuration window.

7.2 SmartMedia™ Properties

This window supplies information about the SmartMedia™ card currently inserted in the MadPlayer™, and also allows you to format a SmartMedia™ Card.

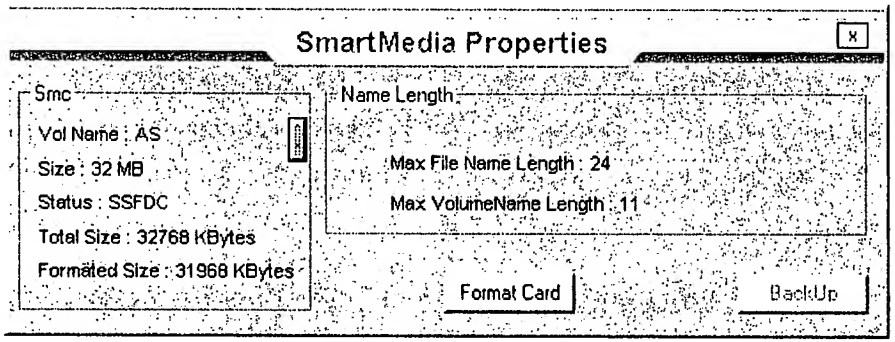


Figure 18 : SmartMedia Properties window.

SmartMedia™ Window functions:

Format Card:
The Format Card button begins the process of formatting the SmartMedia™ Card inserted in the connected MadPlayer™.
Warning: Formatting a SmartMedia™ Card will erase all the existing files stored on that card.

Create a Volume Name:
During the formatting process you can change the volume name of the SmartMedia™ Card by clicking on the small button in the top right corner of the "Smc" area.

Special Requirements (IMPORTANT NOTICE)

SmartMedia™ Cards are high performance devices that require special attention. You should never remove a SmartMedia™ Card from the MadPlayer™ while processing a file transfer, or during any file operation. If MadWare™ is running, you should always "Disconnect," using the Dashboard button, before removing or inserting a SmartMedia™ Card in the connected MadPlayer™. It is also important that you complete any ongoing file transfer before disconnecting the USB cable. If you cannot wait to complete the transfer, use the Abort button to terminate the transfer, and then 'Disconnect' MadWare (or close the MadWare™ program). (See the MadWare™ Installation Guide for complete information on SmartMedia™ Card usage.)

7.3 MadTuner

The MadTuner window creates, edits and stores Radio frequency presets, so that you can program your favorite radio stations to be immediately accessible on the MadPlayer™ radio. Please refer to the MadManual™ User's Guide for complete information about the MadPlayer™ radio.

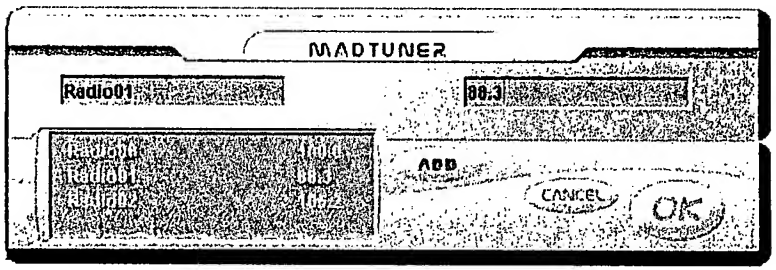


Figure 19 : MadTuner dialog window.

MadTuner window functions:

Adding a Radio Tuner preset:
Click on the Add button and a default Tuner preset is added. (It has a default name of: "Radio00" and a default frequency of: 100.0 MHz.) You can now edit that default preset by following the next step.

Editing a Radio Tuner preset:
Click on a Radio Tuner preset to select it. The Name and Frequency of that selection now appear in the two smaller edit windows. As displayed, you can directly edit both the Name and the Frequency of the selection. To cancel changes press the Cancel button..

Deleting a Radio Tuner Preset:
Click on the preset to be deleted. Then press the Delete (Del) key, or right click on the selected item and choose Delete.

Moving a Radio Tuner Preset:
Click on the item to be moved. Then right click on the selected preset and choose either Up or Down (or use the Down Arrow or Up Arrow keys).

8 MadWorld Mode

MadWorld Mode connects you to the Members Section of the MadWaves web site. Once connected you can download/upload Song or Sample files, as well as access periodic upgrades for the MadPlayer™, or the MadWare™ software program.

Obtaining a Login and Password:
Before you can connect with the MadWaves web site through MadWorld Mode, you must first obtain a Login and Password. To do that you can visit MadWaves.com via the internet and register for a Login and Password, or you can let MadWare™ take you there by choosing MadWaves.com from the Community Internet Links at the bottom of the Dashboard. (See MadWare™ Dashboard.)

8.1 Login

The first time that you open the MadWare™ program, it will attempt to establish an internet connection with the MadWorld site.

Note that this initial connection may take time to be established, depending on your internet connection, and other factors. During the connection process the work area of the Dashboard may remain empty until access has been obtained. [If MadWare™ does not automatically attempt to contact MadWorld, or if you want to restart the process, simply click on the MadWorld Dashboard button.]

Once MadWare™ has successfully connected with the MadWorld Site, the Welcome 2 the MadWorld login page will appear (Figure 20).
Enter the Login and Password that you received by visiting MadWaves.com, and then click on Get In!

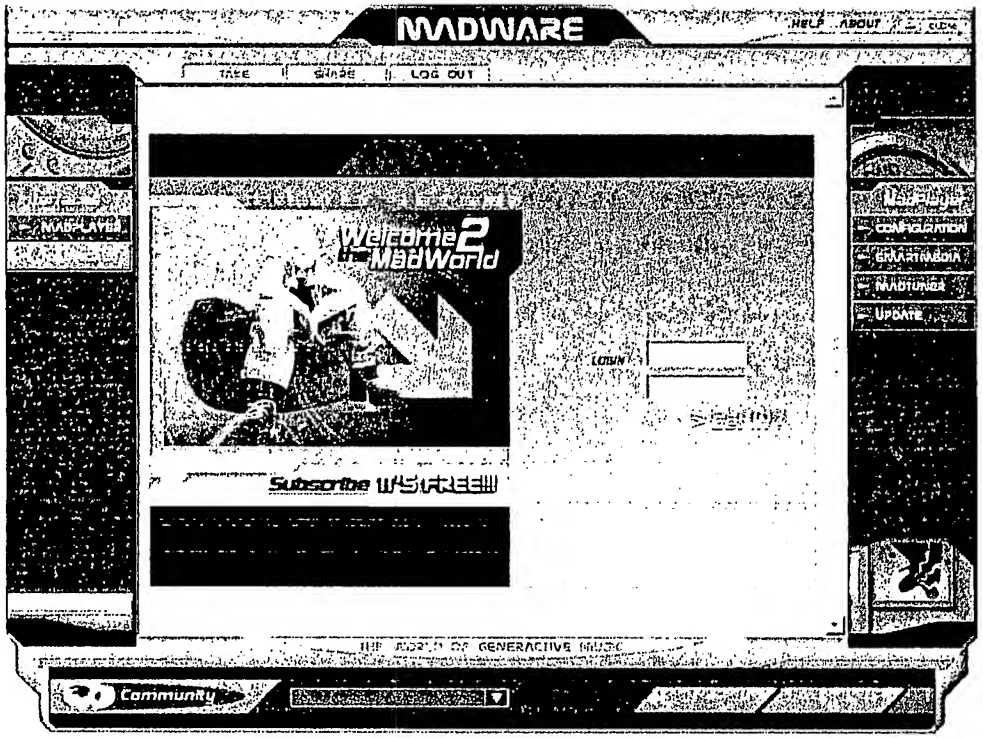


Figure 20 : Welcome 2 the MadWorld screen.

You have now entered the Take & Share interface of the MadWorld, Figure 21.

8.2 Take & Share

It's important to understand that files cannot be directly transferred between the MadWorld site and your connected MadPlayer™. All files must be downloaded to, or uploaded from, the Local Directory, which is your computer hard drive, or some other storage component of your computer. So, if you want to download a song from the MadWorld, you would first download it to the Local Directory and then transfer it to the SmartMedia™ Card of the connected MadPlayer™. If you want to upload a song you've created, you first transfer the MadSong from your MadPlayer's SmartMedia™ Card to the Local Directory, and you then upload the song from the Local Directory to the MadWorld site.

Take (Download) Screen:
The opening (default) screen is the Download screen. To switch to the Share (Upload) screen, click on the Share button (see Figure 21).

Download Steps:
Select a Music Style and click on it to open selections in that style.

Download a Song by clicking on it.

Exit the Take Screen by clicking the LogOut button (to return in the login page), or the Share button (to switch into the share environment), or the MadPlayer™ button (to return in the MadPlayer™ environment).

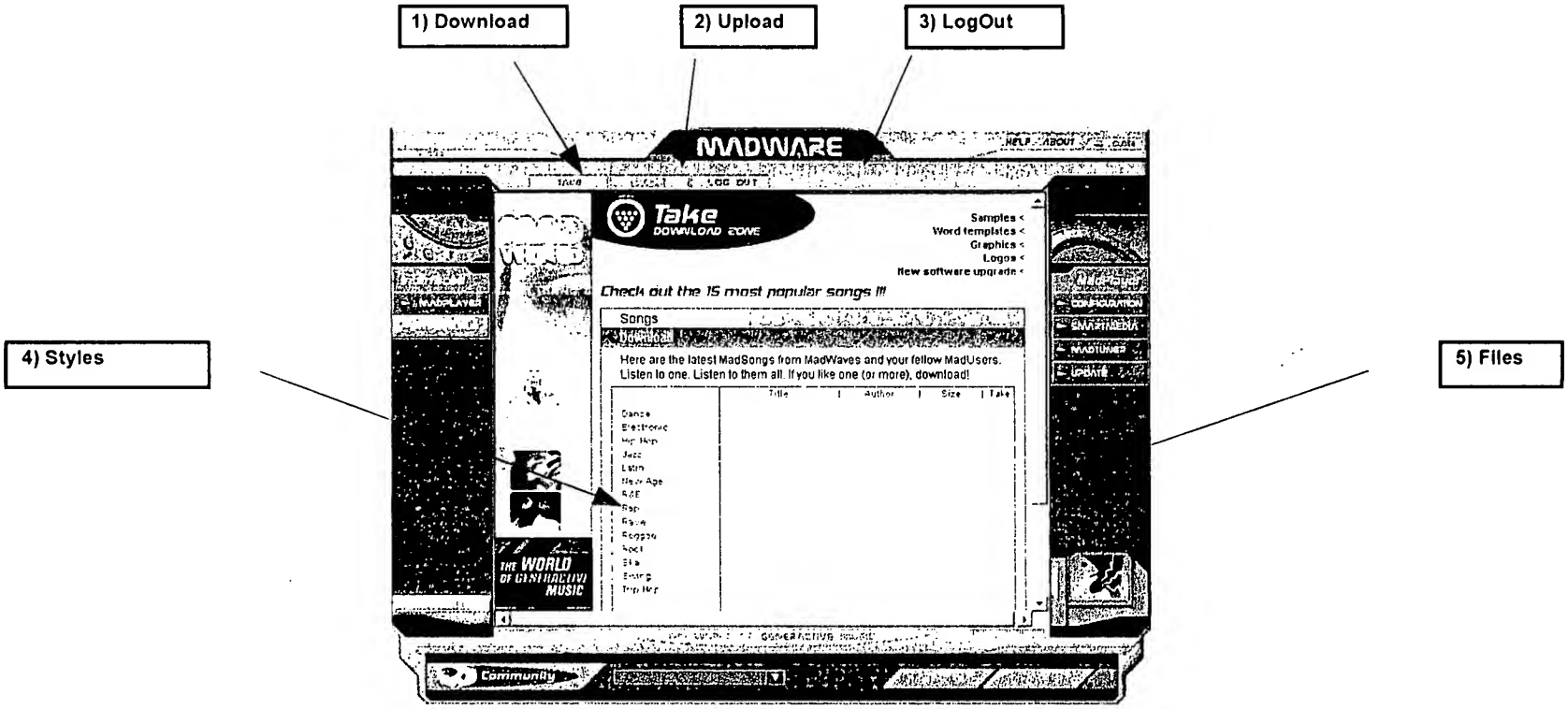


Figure 21 : Downloading files from MadWorld site.

Share (Upload) Screen:

To Upload a MadSong to the Share Zone area of MadWorld, :

- select the type of the file you want to share (Songs, Samples, Word Templates, Graphics)
- fill out the dialog box with the help of the Browse button if needed (to select the file)
- click on Send.

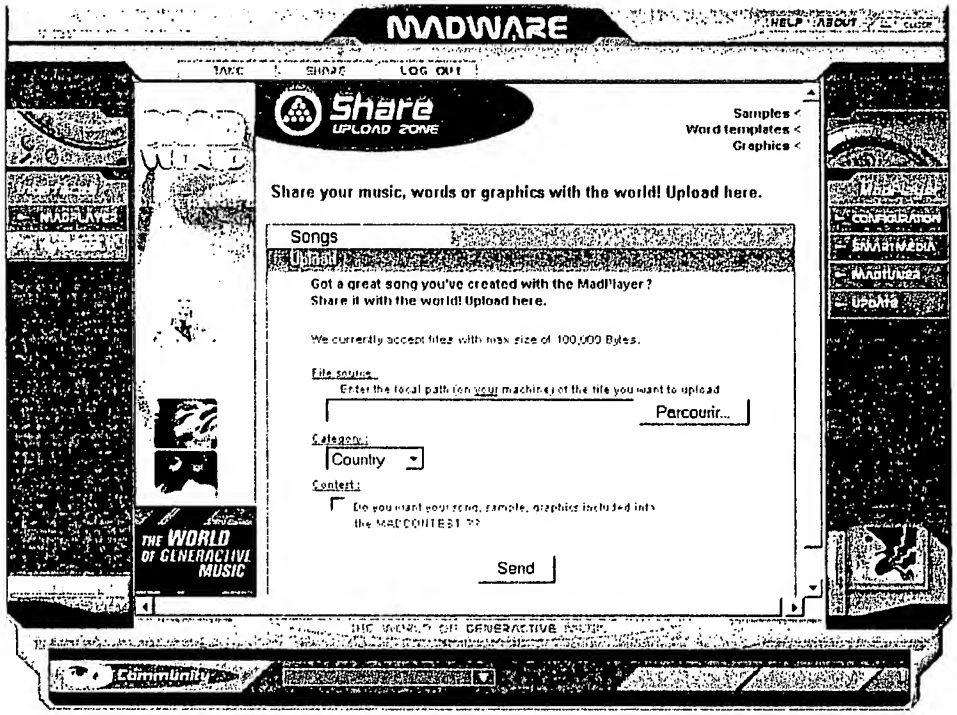


Figure 22 : Sharing files with the MadWorld community